A Location-Independent Direct Link Neuromorphic Interface

Alexander D. Rast, Johannes Partzsch, Christian Mayr, Johannes Schemmel, Stefan Hartmann, Luis A. Plana, Steve Temple, David R. Lester, Rene Schüffny and Steve Furber

***Abstract*—With neuromorphic hardware rapidly moving towards large-scale, possibly immovable systems capable of implementing brain-scale neural models in hardware, there is an emerging need to be able to integrate multi-system combinations of sensors and cortical processors over distributed, multisite configurations. If there were a standard, direct interface allowing large systems to communicate using native signalling, it would be possible to use heterogeneous resources efficiently according to their task suitability. We propose a UDP-based AER spiking interface that permits direct bidirectional spike communications over standard networks, and demonstrate a practical implementation with two large-scale neuromorphic systems, BrainScaleS and SpiNNaker. Internally, the interfaces at either end appear as interceptors which decode and encode spikes in a standardised AER address format onto UDP frames. The system is able to run a spiking neural network distributed over the two systems, in both a side-by-side setup with a direct cable link and over the Internet between 2 widely spaced sites. Such a model not only realises a solution for connecting remote sensors or processors to a large, central neuromorphic simulation platform, but also opens possibilities for interesting automated remote neural control, such as parameter tuning, for large, complex neural systems, and suggests methods to overcome differences in timescale and simulation model between different platforms. With its entirely standard protocol and physical layer, the interface makes large neuromorphic systems a distributed, accessible resource available to all.**

I. Neuromorphic Hardware Systems: A Need for Interfaces

"Neuromorphic" chips, devices that attempt to implement neural networks directly in hardware, are increasing in scale. Various implementations have emerged, ranging from early, small-scale analogue devices following the original model of Carver Mead [1], to more advanced chips implementing specific spiking neural networks [2], through to modern, large-scale systems intended to simulate very large networks with reasonable biological fidelity, possibly at highly accelerated speeds [3]. Recently, the "neuromimetic" branch of neural chips [4] has also begun to explore the configurability dimension, introducing highly flexible chips that can model a wide range of different networks and dynamic models [5]. On both these fronts, neuromorphic chips are clearly moving towards large-scale systems able to simulate substantial fractions of a brain - but the question then becomes how to hook these systems together.

Another branch of neuromorphic engineering, focussed more on practical applications than on neural modelling *per se*, has created interesting sensors designed for particular application scenarios where conventional sensors have inappropriate characteristics or feature sets [6], [7]. The need for fast dynamic response makes neuromorphic processors, whose architecture is naturally matched, an obvious fit for such devices, either to provide "front-end" processing [8], or indeed to perform all the processing on the data. However, this presupposes the availability of a suitable *native* hardware interface, and in actual fact, the hardware interface both at the sensor and the processor side tends to be proprietary. A protocol-level "standard" - AER - exists [9], but it is far from being a standard in the formal sense of the term, and there are fundamental issues still to be resolved at the level of connectors, signal voltages, and other very low-level hardware concerns. If there were an interface, working natively with AER packets, but over a standard hardware medium, and with standard AER protocols, it would thus neatly solve two problems: how to connect neuromorphic sensors to processors, and how to connect neuromorphic processors together.

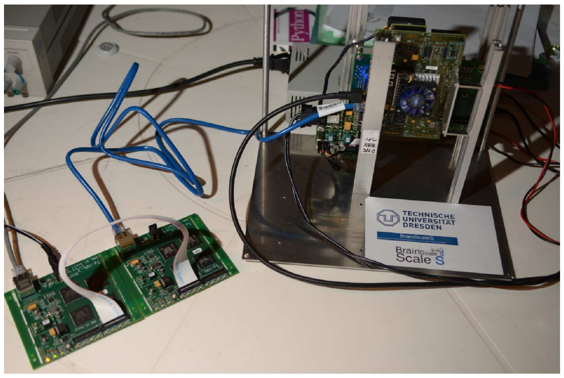
Fig. 1. SpiNNaker to BrainScaleS interface

Some progress has already been made in implementing real multisystem neuromorphic interfaces. The group of Bernabe Linares-Barranco demonstrated as early as 2009 an end-to-end neuromorphic system [10], linking sensors to processors to actuators. Groups at the CapoCaccia neuromorphic workshop have for several years running been creating and refining a general AER standard interface, and in some cases have demonstrated actual hardware [11], [12]. However with signal-level details remaining proprietary, and a data-link design that restricts these interfaces to direct links within a relatively modest distance, such systems are more for bench-level interfaces than multimodel, multisite experiments. They do not address the need to provide remote interfaces between large-scale (and hence nonportable) neuromorphic systems and either real-world sensors or other large-scale systems at other facilities. Here we introduce a neuromorphic interface,

Alexander Rast, Luis A. Plana, Steve Temple, David R. Lester and Steve Furber are with the School of Computer Science, The University of Manchester, Manchester, UK (email: rasta@cs.man.ac.uk). Johannes Partzsch, Christian Mayr, Stefan Hartmann and Rene Schüffny are with Technische Universität Dresden, Dresden, Germany. Johannes Schemmel is with Heidelberg University, Heidelberg, Germany.

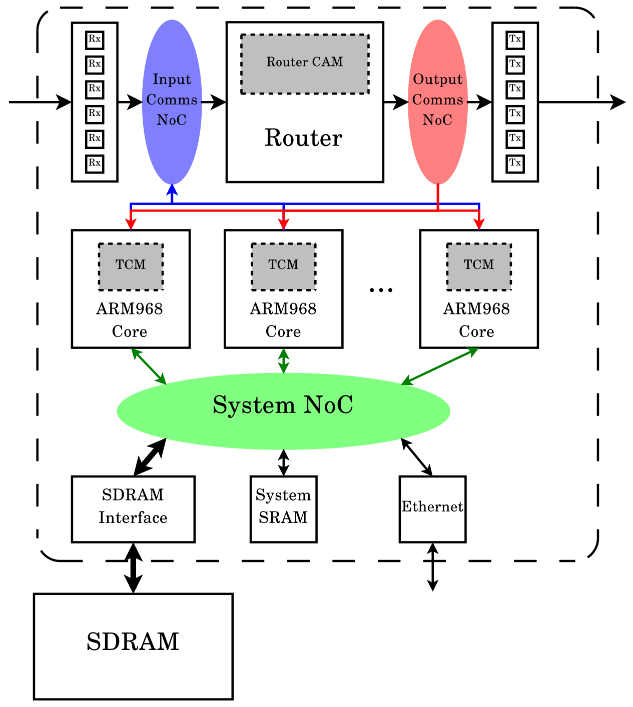

Fig. 2. SpiNNaker Architecture. The dashed box indicates the extent of the SpiNNaker chip. Dotted grey boxes indicate local memory areas.

using standard, networkable UDP protocol, to link two systems: BrainScaleS and SpiNNaker, designed from the outset for large-scale neural simulation, and with easy extensibility to AER-based sensors.

II. SpiNNaker and BrainScaleS: Two Neuromorphic Platforms

To demonstrate a direct neuromorphic interface, we chose two platforms which, while sharing the overall goal of large-scale neural simulation, have very different design goals and specifications that make them truly heterogeneous systems. Here we discuss the architectures of the 2 chips, to provide context for understanding their contrasting system requirements.

A. SpiNNaker

The SpiNNaker chip (fig. 2) is a universal neural network platform designed for real-time simulation. Unlike traditional neuromorphic hardware, SpiNNaker uses an event-driven asynchronous digital design containing programmable processing blocks embedded in a configurable asynchronous interconnect. Like many neuromorphic systems, however, SpiNNaker is specifically designed to operate in real time. There is no central clock; the assumption is that time "models itself", as measured by event rates from the external world. SpiNNaker exploits the large difference between typical electronic response times ($\sim ns - \mu s$) and biological times ($\sim ms$) to support multiple neurons within a single processor, creating an abstract neural hardware platform. This approach embodies the neuromimetic architecture: processors and interconnect are generic and configurable, but have a structure and function optimised for neural computation. The primary features of the neuromimetic architecture are:

Native Parallelism:
There are multiple (18) independent, general-purpose ARM968 processors per device, each operating completely independently (asynchronously) from each other. Each processor has a dedicated private subsystem to support neural functionality. A processor simulates multiple neurons which could be as many as 1000 depending upon a tradeoff between number of neurons per core, model complexity and time resolution (or speed).

Event-Driven Processing:
An external, self-contained, instantaneous signal - an event - drives state change in each processor. which contains a trigger that will initiate or alter the process flow. Each processor node contains a hardware vectored interrupt controller that generates interrupts when an event occurs on the node's support devices. For an external device to interact with SpiNNaker, it must be able to generate and receive events at "real-time-like" speeds.

Distributed Incoherent Memory:
Memory is local rather than global and shared. SpiNNaker processors have access to 2 primary memory resources: their own (64+32)K private "Tightly-Coupled Memory" (TCM) and a per-chip partitioned SDRAM device, neither of which require or have support for coherence mechanisms. Any processor may modify any memory location it can access without notifying or synchronising with other processors. External interfaces cannot rely on complex data-handling routines requiring large amounts of local memory or global memory coherence.

Incremental Reconfiguration:
The structural configuration of the hardware can change dynamically while the system is running. SpiNNaker's communications fabric (the Comms NoC) is a packet-switched asynchronous interconnect using Address-Event Representation to transmit neural signals between processors ([13], [14]). Each chip has a run-time reprogrammable router implementing incremental reconfigurability. An external device, however, must be able to identify itself to this router and network using unique addresses not corresponding to internal neurons, and likewise the configuration must be able to inform SpiNNaker of these external signals.

SpiNNaker appears quite different to the typical neuromorphic platform; where most of the latter implement a fixed physical model, SpiNNaker is instead a substrate for a wide range of different possible models. Additionally, because of its event-driven architecture, SpiNNaker has no fixed time model; it is an abstract-time system where the time model is as programmable as the network model. However, the real-time design objective has implications for the feasible models of time, just as the processors' internal specification has implications for feasible dynamical models of neurons. The critical property is the event rate: feasible time models must generate events at rates that do not overwhelm the processor's event-handling capacity (which itself depends upon the complexity of the neural model).

B. *BrainScaleS Hybrid Neuromorphic System*

The BrainScaleS Hybrid Neuromorphic System (HMF) aims at building a flexible emulation platform for models of biological neural systems based on neuromorphic hardware. Its major building blocks (Fig. 3) are an industry standard compute cluster and multiple neuromorphic hardware modules, interconnected by an OTS Ethernet switch. Each module contains an uncut silicon wafer as well as 12 digital communication subgroups. The wafer integrates 448 HICANN (High Input Count Analog Neural Network) chips [3], implementing mixed-signal circuits capable of modelling 512 neurons ([15]) and 115k synapses. After the fabrication of the HICANN chips additional metal layers are deposited on top of the wafer to create a dense interconnection network as well as the contact pads to interface the wafer to the printed circuit board it is mounted upon.

1) HICANN: The neuromorphic circuits implemented in the HICANN chips operate in continuous time. Spikes generated by the neurons are distributed across the whole wafer surface by the aforementioned interconnection network on the top of the wafer. The system is aimed at emulating medium sized biologically realistic networks (up to about 10^9 synapses) at a very high speed. Some key features to achieve this goal are:

Small Time Constants:
All circuits are tuned to accelerate the emulation by a factor of 10^4 compared to real-time, i.e. the internal time constants (such as the membrane time constant) determining the acceleration factor of the network are 10^4 times shorter than in biology. This makes it unnecessary to bias the neuron and synapse circuits in the deep sub-threshold regime of the transistors while keeping capacitances associated with model time constants reasonably small. Such a speed-up reduces approx. 3 hours of biological time to 1 second of emulation time, making learning experiments and extensive parameter sweeps feasible.

Large Neural Fan-In:
Up to 14k synapses can contribute to the membrane conductance of a single neuron. This is an important prerequisite for biologically plausible models. The average fan-in in the mammalian cortex is assumed to be at least 10k synapses per neuron.

Integrated Calibration Capabilities:
To emulate the different electro-physiological characteristics of nerve cells each neuron circuit can be calibrated using 21 individual and 5 global analog parameters. They are stored in floating-gate memory cells locally to the neuron circuits.

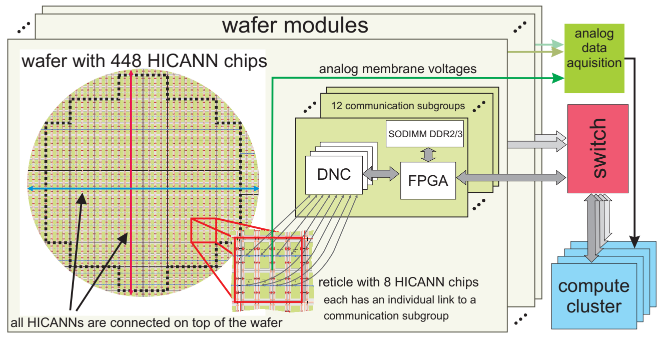
wafer modules
wafer with 448 HICANN chips
analog membrane voltages
analog data aquisition
12 communication subgroups
SODIMM DDR2/3
DNC
FPGA
switch
compute cluster
reticle with 8 HICANN chips each has an individual link to a communication subgroup
all HICANNs are connected on top of the wafer

Fig. 3. Overview of the BrainScaleS Hybrid Neuromorphic System. The central element is an uncut silicon wafer containing 448 neuromorphic HICANN chips interconnected directly on top of the wafer. The total number of synapses per wafer is about 50 million.

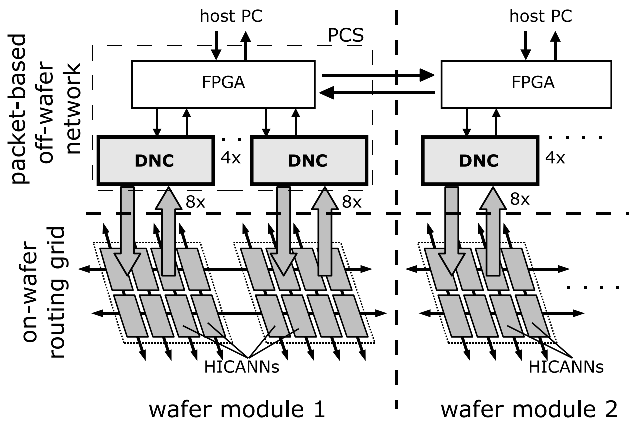

Fig. 4. Components and network connectivity of the BrainScaleS system [17].

2) Communication Network: The high acceleration factor demands a correspondingly high communication bandwidth, not only between individual HICANN chips, but also between different wafer-modules as well as the wafer-modules and the compute cluster. Fig. 4 illustrates the organization of the off-wafer communication network. Eight HICANNs on a wafer, constituting one reticle of the manufacturing process, are connected to a custom- designed digital network chip (DNC) [16], providing high-speed serial interfaces for maximizing throughput to the wafer. This chip also supports pulse handling functionality for realizing biologically realistic constant-delay pulse connections. Four DNCs in turn connect to one FPGA, which provides the host-interface and acts as a network switch for wafer-to-wafer pulse routing in the BrainScaleS system [17]. In the current implementation 12 FPGAs connect to one wafer, limiting the number of usable HICANN chips to 384 out of 448.

Three different communication layers are used throughout the BrainScaleS HMF system:

Connections Local to a Single Wafer: These connections use the links on top of the wafer. The events are transmitted asynchronously in continuous time. 64 neurons are time-multiplexed on a single connection. The encoded neuron number identifies the event's target neurons. The network topology is created by using programmable interconnections on the HICANN chips as well as address decoders located in the synapses.

Neurons to FPGA: Leaving the wafer reduces the available bandwidth substantially. While each HICANN chip has a communication bandwidth of 640 GBit/s, its FPGA connection consists of a full-duplex link with 2 GBit/s. Thus, a packet-based protocol encoding each event with a neuron ID and a time stamp is used, to make better use of the available bandwidth with respect to the real-time links. This encoding also facilitates the routing of events between the different wafers. Furthermore, it allows the implementation of axonal delays by short-term storage of event packets.

FPGA to Compute Cluster: The top communication layer is based on the FPGA's integrated Gbit-Ethernet capabilities. This allows the wafer-module to be integrated directly into any OTS network infrastructure. As used here the FPGA is connected to a host PC for configuration and control using the UDP protocol, different UDP ports distinguishing between core communication and various support functions such as remote reset, low-level configuration, and monitoring. Thus new communication targets, such as the SpiNNaker system used in this work, can be added with little effort by employing an unused port.

III. A Spike Event-Based AER Interface

The AER interface provides a direct communications link between neuromorphic systems supporting an AER protocol - such as BrainScaleS and SpiNNaker. Different chips may have different AER encodings, thus a software layer is responsible for translating spikes from each side into a format decodable by the other side. The interface does not provide any built-in time-domain translation; it issues packets to the target system as quickly as they are received. Likewise, it does not carry additional application information (such as spike payloads or time stamps) that some AER protocols support. This initial version is designed as a demonstrator of suitable techniques for a possible future standard interface for large-scale heterogeneous neural hardware simulation.

A. AER event representation

Since most neuromorphic systems, such as BrainScaleS, use a naked data word to transmit the address, we have defined each AER event as a 32-bit word. Bits 13-0 represent the neuron number. The 16 MSBs represent a hierachical device address, which is prepended to the word and can identify a specific chip or core each simulating up to 8192 neurons. Pad bits are inserted to complete the 32-bit word. For the demonstrator system, an AER packet originating at the BrainScaleS side and targetting SpiNNaker has the format in Table III-A. And in the reverse direction, SpiNNaker-to-BrainScaleS, the packet has the format in Table III-A.

BrainScaleS chip ID	Pad	BrainScaleS Neuron Number
31 16	15 14	13 0

TABLE I. Packet format, BrainScaleS to SpiNNaker

BrainScaleS chip ID	Pad	BrainScaleS Neuron Number
31 16	15 14	13 0

TABLE II. Packet format, SpiNNaker to BrainScaleS

B. Network Protocol

A fundamental design principle of the interface is the use of standard protocols and hardware that present no barriers to adoption by the community. The hardware-level interface is standard IEEE 802.3 Ethernet, and the network-level protocol is UDP running over IP. Each data word contains a padded key (neuron address) in big-endian order. UDP bundles data in frames of up to 65KB, permitting multiple AER packets to be packed into a single frame. However, since the interface issues packets immediately, a given frame can only contain as many packets as arrived simultaneously at the transmit interface. We therefore set the maximum frame size to 256 32-bit data words, or 1024 bytes. As a common Internet protocol, supported both by BrainScaleS and SpiNNaker, UDP makes it possible to communicate between remote locations - simply by plugging the hardware into an Ethernet switch with access to an Internet-accessible router.

The interface translates system-native AER packets into the standard 32-bit AER word, packs simultaneously arriving spike packets into a frame, and issues the frame to the target system. At the target system, the interface unpacks each packet from the frame and injects them directly into the target. By packing spikes into a frame, the interface thus overcomes the potential mismatch between high native spike rates and relatively low UDP frame rates.

IV. Implementation of the Interface

A. Implementation Considerations

As described in section II, BrainScaleS and SpiNNaker have different design goals, leading to some specification differences that require handling. First is native packet formats: BrainScaleS expects 14-bit raw data words, while SpiNNaker expects a packet format with a header and a 32-bit address (and possibly a 32-bit payload, not used here). Thus part of the interface, on both sides, decodes the standard 32-bit AER format into native format. We expect that other chips would likewise involve a similar codec. A second, more subtle consideration is the difference in time scales. Where SpiNNaker was designed to operate in real-time, BrainScales is optimised to run approximately 10^4 times real-time. We placed functions to match the two timescales in a small interface layer residing, in the case of the SpiNNaker system, directly on the main neuromimetic device, and in the case of the BrainScaleS system, in an integrated management FPGA [17].

B. SpiNNaker-Side: The BrainScaleS Interface

Since SpiNNaker processors are entirely general-purpose, it is possible to use any given core on a chip for management and system functions. The standard reference operating model for SpiNNaker designates a "Monitor" core [18] specifically for this purpose. We implemented the BrainScaleS interface as a packet interceptor within the Monitor on the chip connected directly to the Ethernet interface(Fig. 6).

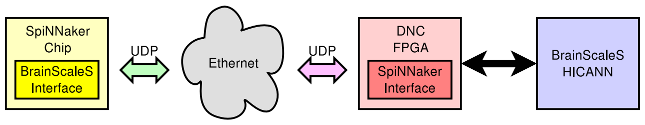

Fig. 5. BrainScaleS-SpiNNaker interface. The SpiNNaker system we used implements the interface directly on the neuromimetic SpiNNaker chip itself. In the BrainScaleS system, since a control FPGA is an integral part of the design [17], we placed the interface on the FPGA, which injects native BrainScaleS packets to the HiCANN chip.

In the outgoing direction, it accepts packets from internal SpiNNaker neurons targetted at BrainScaleS and bundles them into frames. In the incoming direction, it detects packets originating in BrainScaleS, converts them into virtual internal neurons, and injects them onto the SpiNNaker system. To identify BrainScaleS to the system, the SpiNNaker host software registers the BrainScaleS interface as a permanent IP address at start-up. This command both identifies the BrainScaleS IP address to the system (it must be a static IP), and specifies a "virtual chip ID" - the 16-bit address prepended to incoming packets as in III-A. The system also requires a few additional routing table entries to route MC traffic to the interceptor core. Outgoing packets and routing table entries can be automatically configured using the PACMAN tool chain [19]: the user specifies BrainScaleS neurons as a Population, mapped to a virtual core corresponding to the registered address.

During operation, the interceptor on the incoming direction:

1) Captures and buffers incoming UDP packets from the BrainScaleS ID.
2) Schedules a packet-issuing task which dequeues the buffer.
3) Issues multicast packets containing the origin virtual chip/core ID from each 32-bit word.

And in the outgoing direction:

1) Intercepts MC traffic with source IDs matching the range to be sent to the BrainScaleS interface.
2) Strips off the unused high bits.
3) Assembles the packets into UDP frames.
4) Issues the frames to the BrainScaleS Ethernet port.

To cope with the difference in time scales, on the SpiNNaker side, we have implemented a packet repeater. While a given neuron in SpiNNaker generates only one packet to output to the Monitor, the interceptor generates multiple packets for the same neuron, simulating a population. The number of duplicate packets and their timing can be configured via parameters at system start-up.

C. BrainScaleS-side: The SpiNNaker interface

Handling the pulse communication on the BrainScaleS side is done completely in the system's FPGAs. These are responsible for address decoding and pulse processing. To bridge the gap in the operating timescales, incoming pulses from SpiNNaker are multiplied between 100 and 1000 times at a frequency of approx. 100kHz (corresponding to 10Hz in

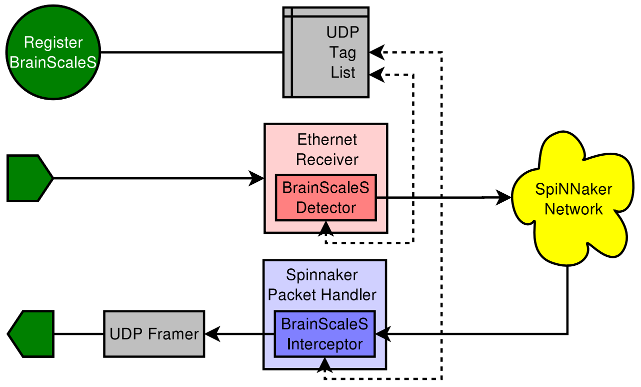

Fig. 6. SpiNNaker-side BrainScaleS interface. The diagram shows the internal processes active on the Monitor. "Register BrainScaleS" is an external command originating in the host. The SpiNNaker network is its own internal AER asynchronous interconnect.

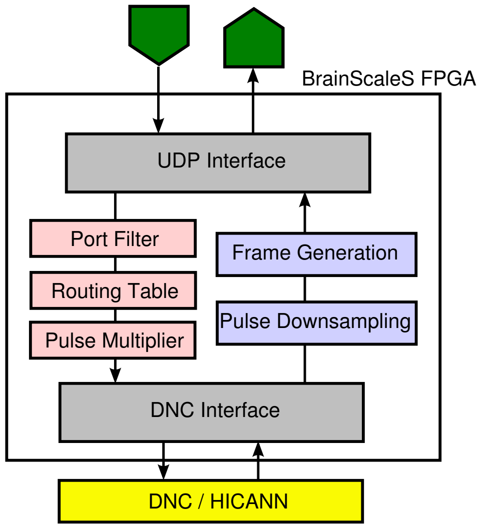

Fig. 7. BrainScaleS-side SpiNNaker interface, showing the different pulse processing steps for incoming and outgoing traffic.

accelerated time). This partially matches the spike rate of the incoming pulse stream to the speed-up of the BrainScaleS system. In effect, the accelerated neurons are sufficiently stimulated even with spike trains with plausible real-time rates. Spikes generated by the neurons on the HICANNs are likewise downsampled by a factor of 100 to 1000 to partially match the mean rates in the BrainScaleS-to-SpiNNaker direction.

Fig. 7 shows the SpiNNaker interface blocks on the BrainScaleS FPGA. As per sec. II-B, the BrainScaleS FPGA uses UDP ports to differentiate between different types of communication targets. To interface with SpiNNaker, a port filter selects incoming UDP frames from the SpiNNaker system. These frames are forwarded to a configurable routing table, which translates the incoming 32bit words into 14bit local addresses, specifying the target HICANN and virtual source neuron address. The translated pulses are then inserted in a ring buffer, which realises the pulse multiplication by continuously looping over all buffer entries and generating spikes with the entry's address. Pulses generated by neurons

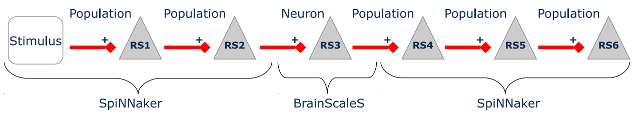

Fig. 8. Structure of the synfire chain experiment and distribution among the two systems.

on the HICANNs are sent to a downsampling unit that only forwards every n-th spike to a frame generation module, which bundles simultaneous spikes and sends them via the UDP interface to the SpiNNaker side.

V. Dual-System, Multi-Time Domain Simulation

In order to verify the interface, we tested the system both with a local connection, the two systems side-by-side and connected directly through an Ethernet cable (as in fig. 1), and over the Internet, each system residing in a remote location. We configured a standard benchmark test that both systems could execute, and that represents a reasonably realistic application scenario. Results verify the correct operation of the interface.

A. Test Experiment

For testing the combined systems in a reasonable experiment, a simplified version of a synfire chain [20] was used. This network is suited for testing the interface for two reasons. First, it is a feed-forward network, so that the delay of the interface, being very different for local or remote setup, has no influence on the network behaviour, as would be the case for a recurrent network. Second, the synfire chain is selective for the relative spike timings. More precisely, if spikes are spread in time, they do not trigger a transmission through the chain, while short burst of spikes are safely transmitted. Reproducing this behaviour is a challenging task for a combined system operating at different timescales.

The test bench is a script written in the PyNN [21] network description language, executed on the SpiNNaker system. Neurons on the SpiNNaker side are configured so that a given neuron in the chain will fire whenever it receives a spike. As shown in Fig. 8, the chain contains 6 populations of neurons, of which SpiNNaker simulates five. We then inserted BrainScaleS to simulate the 2nd population, to complete the chain. On the BrainScaleS system, one neuron with ten inputs was utilized, together with the pulse processing described in Sec. IV-C. The BrainScaleS part with its accelerated time base essentially operates as a coincidence detector, generating output spikes only if enough inputs are spiking in a short time window. The SpiNnaker system was configured to run in real-time, with a simulation time step of 1 ms. The listing below gives the PyNN script.

```
#!/usr/bin/python
import pyNN.spiNNaker as p

p.setup(timestep=1.0, min_delay = 1.0, max_delay = 8.0, db_name=’synfire.sqlite’)

n_pop = 6 #60
nNeurons = 100 #100

p.get_db().set_number_of_neurons_per_core(’IF_curr_exp’, nNeurons)

rng = p.NumpyRNG(seed=28374)
rng1 = p.NumpyRNG(seed=12345)
```

```
delay_distr = p.RandomDistribution(’uniform’, [5,10], rng)
weight_distr = p.RandomDistribution(’uniform’, [0,2], rng1)

v_distr = p.RandomDistribution(’uniform’, [-55,-95], rng)

v_inits = []
for i in range(nNeurons):
    v_inits.append(v_distr.next())

cell_params_lif_in = {’tau_m’:32, ’v_init’:-80, ’v_rest’:-75, ’v_reset’:-95,
’v_thresh’:-55, ’tau_syn_E’:5, ’tau_syn_I’:10, ’tau_refrac’:20, ’i_offset’:1}

cell_params_lif = {’tau_m’:32, ’v_init’:-80, ’v_rest’:-75, ’v_reset’:-95,
’v_thresh’:-55, ’tau_syn_E’:5, ’tau_syn_I’:10, ’tau_refrac’:5, ’i_offset’:0}

populations = list()
projections = list()

weight_to_spike = 20

for i in range(n_pop):
    if i == 0:
        populations.append(p.Population(nNeurons, p.IF_curr_exp,
        cell_params_lif_in, label=’pop_%d’ % i))
        populations[i].randomInit(v_distr)
        BSpopulation = p.Population(nNeurons, p.IF_curr_exp, cell_params_lif,
        label=’BrainScaleS_Dummy’)
        BSpopulation.set_mapping_constraint({’x’:2, ’y’:0, ’p’:1})
    else:
        populations.append(p.Population(nNeurons, p.IF_curr_exp, cell_params_lif,
        label=’pop_%d’ % i))
    if i == 1 or i > 2:
        print i
        projections.append(p.Projection(populations[i-1], populations[i],
        p.OneToOneConnector(weights=weight_to_spike, delays=10)))
    else:
        projections.append(p.Projection(BSpopulation, populations[i],
        p.OneToOneConnector(weights=weight_to_spike, delays=10)))
    populations[i].record_v()
    populations[i].record(to_file=False)

projections.append(p.Projection(populations[1], BSpopulation,
p.OneToOneConnector(weights=weight_to_spike, delays=10)))

p.run(10000)
p.end()
```

B. Dual-System Testing

The first tests we ran placed the systems side-by-side in a single location, for direct observability and mitigation of possible networking issues. Indeed, we observed that for some network configurations, a remote setup could experience connectivity problems when the systems on either side were separated by an intervening firewall. Obviously in a direct connection situation this was not a concern.

In the second series of tests, the two systems were in separate locations - the SpiNNaker system at the University of Manchester, the BrainScaleS system at TU Dresden. We also configured the SpiNNaker system remotely from Dresden, using `ssh` to connect to the SpiNNaker local host. For this test we also optimised the BrainScaleS parameters for a selective response, setting the pulse multiplication to 200 and the downsampling to 100.

Figs. 9 and 10 show the results for the remote operation, the first one for a narrow initial spike burst, the second one for a wider distribution. Both results show the successful combined, remote operation of both systems. Note that in both cases, the timing of received spikes from the BrainScaleS system on SpiNNaker is much tighter than the originating spikes, because of the higher native speed of the HiCANN chip, and the absence of any time-domain translation.

The comparison of both figures also shows that the whole chain is indeed selective: For a narrow initial spike distribution, transmission is stable and the populations following the BrainScaleS one are rather forced to spike. In contrast, when choosing a slightly wider initial spike burst, transmission starts to fade out and get unreliable. Even wider distributions would result in an almost complete shut down of activity after the BrainScaleS part.

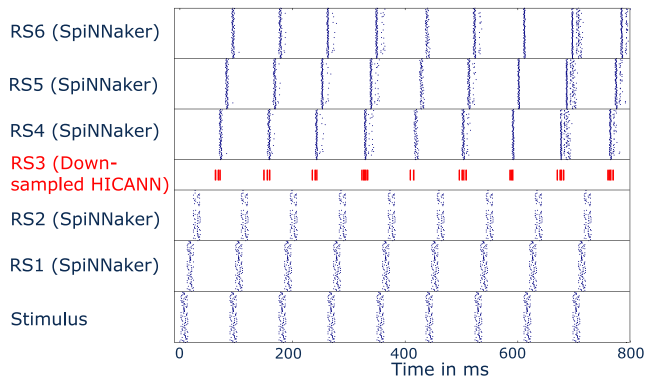

Fig. 9. Synfire chain, combined BrainScaleS-SpiNNaker system, narrow initial stimulation

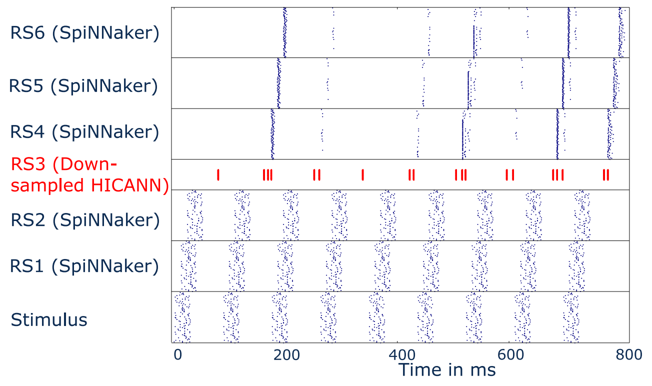

Fig. 10. Synfire chain, combined BrainScaleS-SpiNNaker system, wide initial stimulation

The successful execution of these tests and the improved output results verify the functionality of the interface and confirm its suitability for active multisite simulations.

VI. Discussion: A Networkable Standard Neuromorphic Interface

Developing and running an interface for two real-world systems in a real-time simulation scenario makes obvious several points about both the potential and the practicalities of neuromorphic interfaces.

A. Dynamic System Tuning

A direct interface between the SpiNNaker and BrainScaleS system makes it possible to observe interactively the effects of different models, and different parameters, during multi-system simulation. One way this was of immediate use was in tuning the BrainScaleS parameters for the synfire chain model. Taking advantage of the direct observability available by running the model interactively, it is possible to follow a simple iterative configure-run-examine results cycle to tune the BrainScaleS chip. A similar model can be applied to other processes, either model-specific or hardware-specific, that require tuning, and because the interface allows for remote access, either hardware system can be configured in this way from a model running on either side, or more generally between any group of remotely-connected devices. Such manual configuration would be the simplest way to perform system tuning, but it is not unreasonable to consider a more advanced method: using one system to configure the other automatically, in the manner of a neuromodulatory path, allowing for dynamic-real-time system tuning. This is one direction we are looking at for the future.

B. Next Steps: Future Work

Dynamical tuning using neuromodulation is one example of an advanced, multi-system application we are investigating as part of the next steps. This approach is particularly relevant in the SpiNNaker-to-BrainScaleS system: inherently, SpiNNaker runs in a slow time domain, relative to BrainScaleS, which makes it a natural fit to model slow neuromodulatory processes on a model running at high speeds within the BrainScaleS system. More immediately, we are working to integrate a silicon retina into the system using the same interface, demonstrating the use of the approach for remote sensors or actuators as well as connections between cortical-like processors modelling cognitive processes. In addition we will be investigating more thoroughly the behaviour of simulations across heterogeneous time domains and developing techniques for time domain bridging. These directions become possible because of the basic work here to create a standard interface that permits remote connectivity between different neuromorphic systems.

C. What Should A Neuromorphic Interface Look Like?

The question of neuromorphic interfaces has been engaging the research community for several years now. Very early on in the development of the BrainScaleS-SpiNNaker interface, it became clear that the only practicable approach would be to use a standard Ethernet connection - because this is one of the few built-in components that almost every device supports. With this research, we propose several key points about what such a neuromorphic interface should look like:

Remote Accessibility:
Two devices need not be physically proximate to communicate.
Industry Standard Protocol:
The interface uses available public protocols with existing support.
Off-the-Shelf Physical Interface:
The physical layer, connectors, and cabling use existing hardware standards .
Address-Event-Representation:
The neural layer interface is a spike-based AER protocol with simple packet format.

This style of universal, globally accessible interface is suitable as a standard for the upcoming generation of neuromorphic systems: large, multisite, heterogeneous platforms.

VII. Conclusions

The interface we have implemented between the BrainScaleS and the SpiNNaker systems bridges two neural hardware platforms that have been designed from the outset to be large-scale systems. Such systems, sited in a fixed location and accessible to modellers through public Internet interfaces will become an increasingly typical pattern in neuromorphic hardware, as systems scale towards brain-scale simulation. However, the question of interfacing to such large systems seems, in the past, to have been thought through mostly with respect to pre-built simulations targetted at a specific platform, in the mode of a traditional HPC computing model. Such a model is probably unrealistic in the case of neuromorphic systems, which may need to interface to sensors and actuators (themselves quite probably neuromorphic chips), or in fact other neuromorphic systems simulating other high-level cognitive processes. An interface such as we have developed, that permits direct communications over the Internet via a private channel, makes it possible to hook in these devices directly into a large system, as if they were beside it.

Furthermore, research either into brain function or neural computing models at large scales usually takes the form of collaborations between multiple research groups, typically having several computing platforms. Immediate benefits like cross-system parameter tuning, as we observed in our experiments, are a simple way a remote neuromorphic interface enhances the value of multiple platforms. Far more significantly, however, different neuromorphic chips will have different strengths: perhaps multi-model simulation in one case, single-model accuracy for another, high speed for a third. A large-scale neural simulation may need all these capabilities and more, for example to examine one area in high detail while retaining an abstract model for other necessary processes in the entire simulation. The interface we have developed allows these groups to use multiple neuromorphic platforms as a single, integrated heterogeneous system rather than being forced to operate them in isolation.

Acknowledgements

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007- 2013) under grant agreement no. 269921 (BrainScaleS). The Spinnaker project is supported by the Engineering and Physical Sciences Research Council, partly through the Advanced Processor Technologies Platform Partnership at the University of Manchester, and also by ARM.

References

[1] D. W. Tank and J. J. Hopfield, "Simple "Neural" Optimization Networks: an A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit," vol. 33, no. 5, pp. 533–541, May 1986.

[2] G. Indiveri, E. Chicca, and R. Douglas, "A VLSI Array of Low-Power Spiking Neurons and Bistable Synapses With Spike-Timing Dependent Plasticity," *IEEE Trans. Neural Networks*, vol. 17, no. 1, pp. 211–221, Jan. 2006.

[3] J. Schemmel, D. Brüderle, A. Grübl, M. Hock, K. Meier, and S. Millner, "A Wafer-Scale Neuromorphic Hardware System for Large-Scale Neural Modeling," in *Proc. 2010 IEEE Int'l Symp. Circuits and Systems (ISCAS2010)*, 2010, pp. 1947–1950.

[4] S. Renaud, J. Tomas, Y. Bornat, A. Daouzli, and S. Saïghi, "Neuromimetic ICs With Analog Cores: An Alternative for Simulating Spiking Neural Networks," in *Proc. IEEE Int'l Symp. Circuits and Systems (ISCAS2007)*, 2007, pp. 3355–3358.

[5] A. Rast, F. Galluppi, S. Davies, L. Plana, C. Patterson, T. Sharp, D. Lester, and S. Furber, "Concurrent heterogeneous neural model simulation on real-time neuromimetic hardware," *Neural Networks*, vol. 24, no. 9, pp. 961–978, Nov. 2011.

[6] C. Posch, D. Matolin, and R. Wohlgenannt, "A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS," *IEEE J. of Solid-State Circuits*, vol. 46, no. 1, pp. 259–275, Jan. 2011.

[7] V. Chan, S.-C. Liu, and A. van Schaik, "AER EAR: A matched silicon cochlea pair with address event representation interface," *IEEE Trans. Circuits and Systems 1: Fundamental Theory and Applications*, vol. 54, no. 1, pp. 48–59, Jan. 2007.

[8] L. Camuñas-Mesa, A. Acosta-Jiménez, T. Serrano-Gotarredona, and B. Linares-Barranco, "Fully Digital AER Convolution Chip for Vision Processing," in *Proc. 2008 IEEE Int'l Symp. Circuits and Systems (ISCAS2008)*, 2008, pp. 652–655.

[9] J. Lazzaro, J. Wawrzynek, M. Mahowald, M. Silviotti, and D. Gillespie, "Silicon Auditory Processors as Computer Peripherals," *IEEE Trans. Neural Networks*, vol. 4, no. 3, pp. 523–528, May 1993.

[10] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gómez-Rodríguez, L. Camuñas-Mesa, R. Berner, M. Rivas-Pérez, T. Delbrück, S.-C. Liu, R. Douglas, P. Häfliger, G. Jiménez-Moreno, A. Civit-Ballcels, T. Serrano-Gotarredona, A. J. Acosta-Jiménez, and B. Linares-Barranco, "CAVIAR: A 45k neuron, 5M synapse, 12G connects/s AER hardware sensory-processing-learning-actuating system for high-speed visual object recognition and tracking," *IEEE Trans. Neural Networks*, vol. 20, no. 9, pp. 1417–1438, Sep. 2009.

[11] E. Chicca, A. M. Whatley, P. Lichtsteiner, V. Dante, T. Delbruck, P. del Giudice, R. J. Douglas, and G. Indiveri, "A Multichip Pulse-Based Neuromorphic Infrastructure and Its Application to a Model of Orientation Sensitivity," *IEEE Trans. Circuits and Systems*, vol. 54, no. 5, pp. 981–993, May 2007.

[12] D. B. Fasnacht, A. M. Whatley, and G. Indiveri, "A Seral Communication Infrastructure for Multi-Chip Address Event Systems," in *Proc. 2008 IEEE Int'l Symp. Circuits and Systems (ISCAS2008)*, 2008, pp. 648–651.

[13] L. A. Plana, S. B. Furber, S. Temple, M. M. Khan, Y. Shi, J. Wu, and S. Yang, "A GALS Infrastructure for a Massively Parallel Multiprocessor," *IEEE Design & Test of Computers*, vol. 24, no. 5, pp. 454–463, Sep.-Oct. 2007.

[14] M. M. Khan, D. Lester, L. Plana, A. Rast, X. Jin, E. Painkras, and S. Furber, "SpiNNaker: Mapping Neural Networks Onto a Massively-Parallel Chip Multiprocessor," in *Proc. 2008 Int'l Joint Conf. Neural Networks (IJCNN2008)*, 2008, pp. 2849–2856.

[15] S. Millner, A. Grübl, K. Meier, J. Schemmel, and M.-O. Schwartz, "A VLSI implementation of the adaptive exponential integrate-and-fire neuron model," in *Advances in Neural Information Processing Systems 23*, J. Lafferty *et al.*, Eds., 2010, pp. 1642–1650.

[16] S. Scholze, H. Eisenreich, S. Höppner, G. Ellguth, S. Henker, M. Ander, S. Hänzsche, J. Partzsch, C. Mayr, and R. Schüffny, "A 32 GBit/s communication SoC for a waferscale neuromorphic system," *Integration, the VLSI Journal*, vol. 45, no. 1, pp. 61–75, 2011.

[17] S. Scholze, S. Schiefer, J. Partzsch, S. Hartmann, C. Mayr, S. Höppner, H. Eisenreich, S. Henker, B. Vogginger, and R. Schüffny, "VLSI implementation of a 2.8GEvent/s packet based AER interface with routing and event sorting functionality," *Frontiers in Neuroscience*, vol. 5, no. 33, p. 28, 2011.

[18] S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown, "Overview of the SpiNNaker system architecture," *IEEE Trans. Computers*, vol. PP, no. 99, 2012.

[19] F. Galluppi, A. Rast, S. Davies, and S. Furber, "A General-purpose Model Translation System for a Universal Neural Chip," in *Proc. 17th Int'l. Conf. on Neural Information Processing (ICONIP 2010)*. Springer-Verlag, 2010, pp. 58–65.

[20] J. Kremkow, L. Perrinet, G. Masson, and A. Aertsen, "Functional consequences of correlated excitatory and inhibitory conductances in cortical networks," *Journal of Computational Neuroscience*, vol. 28, pp. 579–594, 2010.

[21] A. P. Davison, D. Brüderle, J. Eppler, J. Kremkow, E. Muller, D. Pecevski, L. Perrinet, and P. Yger, "PyNN: a common interface for neuronal network simulators," *Frontiers in Neuroinformatics*, vol. 2, no. 11, Jan. 2009.