Scalable Hierarchical Network-on-Chip Architecture for Spiking Neural Network Hardware Implementations

Snaider Carrillo, *Member*, *IEEE*, Jim Harkin, Liam J. McDaid, Fearghal Morgan, Sandeep Pande, Seamus Cawley, and Brian McGinley

Abstract—Spiking neural networks (SNNs) attempt to emulate information processing in the mammalian brain based on massively parallel arrays of neurons that communicate via spike events. SNNs offer the possibility to implement embedded neuromorphic circuits, with high parallelism and low power consumption compared to the traditional von Neumann computer paradigms. Nevertheless, the lack of modularity and poor connectivity shown by traditional neuron interconnect implementations based on shared bus topologies is prohibiting scalable hardware implementations of SNNs. This paper presents a novel hierarchical network-on-chip (H-NoC) architecture for SNN hardware, which aims to address the scalability issue by creating a modular array of clusters of neurons using a hierarchical structure of low and high-level routers. The proposed H-NoC architecture incorporates a spike traffic compression technique to exploit SNN traffic patterns and locality between neurons, thus reducing traffic overhead and improving throughput on the network. In addition, adaptive routing capabilities between clusters balance local and global traffic loads to sustain throughput under bursting activity. Analytical results show the scalability of the proposed H-NoC approach under different scenarios, while simulation and synthesis analysis using 65-nm CMOS technology demonstrate high-throughput, low-cost area, and power consumption per cluster, respectively.

Index Terms—Interconnection architecture, network-on-chip, neurocomputers, real-time distributed, spiking neural networks

1 Introduction

Spiking neural networks (SNNs) offer a closer approach to modeling biological neurons than previous artificial neural network models [1]. SNNs attempt to emulate information processing in the mammalian brain based on massively parallel arrays of neurons that communicate via spike events. SNNs use the timing of spikes, the network topology, and synaptic weights to process information.

A spiking neuron consists of a cell body (soma) that possesses many input branches called dendrites, which carry information from other neurons. The neuron output (axon) communicates information to other neurons in the form of spikes. Spikes are transmitted between neurons via weighted connections called synapses. A neuron generates an output spike when its post-synaptic response exceeds a firing threshold value. This repeated process enables tasks such as classification or pattern recognition to be performed. Moreover, SNNs exhibit the ability to quickly adapt to failing synapses or neurons providing fault-tolerant capabilities.

The aforementioned attributes and the ability of SNNs to provide a good solution in partially observable environments make SNNs suitable for implementing embedded neuromorphic circuits [2] and control applications [3], with high parallelism and low power consumption compared to the traditional von Neumann computer paradigms [4]. Consequently, one key research goal is to harness this efficiency and build artificial neural systems that can emulate the information processing principles of the brain.

For large SNNs, hardware implementations offer superior execution speed compared to sequential software approaches due to the inherent parallelism of hardware [5]. However, traditional direct neuron-to-neuron interconnection based on a shared bus topology, similar to the one shown in Fig. 1, is not scalable since the number of bus lines required to interconnect the neurons is proportional to the number of neurons located in the presynaptic layer, times the number of neurons located in postsynaptic layer. Therefore, this nonlinear increase in neural connectivity is too significant to be implemented using a dedicated point-to-point scheme.

In addition, even though spike event rates are slower (15 spikes per second) [6] compared to the average data rate shown by modern multiprocessor systems-on-chip (MPSoC), typically in the range of gigabits per second; the amount of neurons to be connected is at least 10^3 times larger than the amount of processing elements that need to be interconnected on current MPSoC platforms [7]. Hence, theses aforementioned constraints make the deployment of brain-like

- S. Carrillo, J. Harkin, and L.J. McDaid are with the Intelligent Systems Research Centre, School of Computing and Intelligent Systems, University of Ulster, Derry BT48 7JL, Northern Ireland, United Kingdom. E-mail: carrillo_lindado-s@email.ulster.ac.uk, {jg.harkin, lj.mcdaid}@ulster.ac.uk.
- F. Morgan, S. Pande, S. Cawley, and B. McGinley are with the Bio-Inspired Electronics and Reconfigurable Computing Research Group, Electrical and Electronic Engineering, National University of Ireland, Galway, Republic of Ireland. E-mail: {fearghal.morgan, sandeep.pande, s.cawley6, brian.mcginley}@nuigalway.ie.

Manuscript received 29 Mar. 2012; revised 1 Sept. 2012; accepted 25 Sept. 2012; published online 3 Oct. 2012.
Recommended for acceptance by M.E. Acacio.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2012-03-0325.
Digital Object Identifier no. 10.1109/TPDS.2012.289.

1045-9219/13/$31.00 © 2013 IEEE Published by the IEEE Computer Society

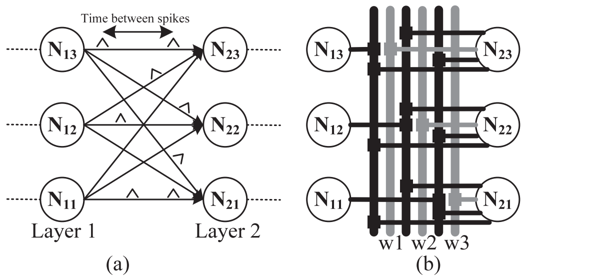


Fig. 1. (a) shows an example of a 2×3 neural network and (b) shows its implementation using a bus-based interconnect scheme, where the dark bus lines are used to send the spike events, whereas the gray bus lines provide the different synaptic weight.

embedded systems a challenging on-chip interconnect problem, where a balance between scalability and biological real-time requirements needs to be achieved. This scalability issue, therefore, necessitates a new full-custom hardware interconnect architectures which can address this scalability issue.

1.1 Background and Motivation

The network-on-chip (NoC) paradigm was introduced in [8], [9], [10], as a promising solution to solve the on-chip communication problems experienced in MPSoC. In general, NoC architectures are composed of a set of shared cores or processing elements, routers, and links, which are arranged in a specific topology depending on the application. In the context of SNNs, the organization of the NoC infrastructure can be viewed as follows: The processing elements refers to the spiking neuron models attached to NoC routers, the NoC channels are analogous to the synapses of neurons, and the NoC topology in this case refers to the way those neurons are interconnected across the network.

Although previous work [11], [12], [13], [14] has provided the foundations for using NoC as an interconnect fabric for SNNs, their interconnection strategies or flat topologies, i.e., one-to-one correspondence between neurons and routers, make it difficult to achieve high scalability with low power consumption. The latter is of vital importance because a high percentage of the power consumed by NoC-based SNN hardware implementations is produced by the interconnection fabric (i.e., routers, index tables, etc.), where the neuron model typically has a power consumption approximately six orders of magnitude smaller than that of the interconnect fabric [15].

Previous work [16], [17], [18] has shown that SNNs can exhibit short communication paths, i.e., high communication locality between neurons, allowing a group of neurons allocated within the same region to share the same incoming spike events. With this motivation, this paper explores hierarchical NoCs (H-NoCs). H-NoCs can be described as an on-chip fabric, where virtual regions or facilities are ranked in different communication levels to process either local or global traffic, as a way to exploit the concept of region-based routing [19].

Recently, H-NoCs have seen success in addressing scalability and offering low power footprints in other large-scale applications such as chip multiprocessor (CMP) systems [20], SoC platforms, and video image processing [21]. In the context of SNNs, these virtual regions or facilities can be used to create clusters of neurons by isolating neurons with local and global connections. Hence, this multilevel NoC structure allows the variation in local and global connections between neurons to be accommodated for scalable implementations.

A recent theoretical evaluation of a hierarchical 2D-mesh topology for neural network hardware that uses an embedded processor to emulate neuron behavior [22] shows the theoretical benefits of grouping neurons based on a hierarchical organization. Nevertheless, the evaluation is constrained only to mesh topology and does not consider combined topologies for H-NoC.

1.2 Contribution of This Work

This paper presents an advanced hierarchical NoC architecture for SNN implementations by combining star-mesh topologies to better exploit the one-to-many multicast communication between neurons in the networks. The main building block for the proposed H-NoC architecture is the *cluster facility*, where a group of neurons are interconnected based on a hierarchical structure, using an array of low and high-level NoC routers to support the local (*intracluster facility*) and global (*intercluster facility*) connections between neurons.

The main contributions of this paper include:

- A hardware implementation of a novel H-NoC architecture for SNN implementations.
- A specialized array of NoC router architectures combined with traffic compression and congestion mechanisms.
- Supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2012.289, to show detailed analysis of SNN traffic, validation of throughput, scalability, and area/power footprint for the H-NoC.

The proposed H-NoC architecture is validated based on an RTL-level implementation, while area and power analysis are performed using 65-nm CMOS technology. Results show significant scalable characteristics such as high throughput and low power/area footprint that make it suitable for large-scale SNN-embedded implementations.

A preliminary version of this paper appeared in [23]; however, this paper presents new sections to provide a detailed overview of the complete H-NoC architecture and fundamental results, while supplementary material, available online, shows additional results and simulations to demonstrate its scalability. The remainder of this paper is organized as follows: Section 2 presents related work. Section 3 introduces the proposed H-NoC architecture, describing its three on-chip communication levels and the traffic compression technique for SNNs. Section 4 presents results from the evaluation of the hierarchical approach and traffic compression in terms of scalability, throughput and area/power footprint. Section 5 concludes the paper.

2 Related Work

This section presents a brief overview of previous works. In general, these works can be classified into high-performance computer (HPC) and full-custom design categories, according to the technology used to accommodate the neural models. Nevertheless, this section only focuses on work that has shown practical demonstration of scalability for large-scale SNN hardware implementations. A detailed review of artificial neural network hardware implementations can be found in [24].

2.1 High-Performance Computer Approaches

Nowadays, the Blue Brain project [25] is the flagship of the HPC trend regarding SNNs. Currently, the Blue Brain project is able to simulate up to 10^8 simple neurons or up to 10^4 very complex neurons, (in this scenario, the "simplicity" depends of the ion channel level details, as well as local and global synaptic plasticity rules defined for each neuron). The simulation environment is supported on the IBM Blue Gene/L, a HPC that offers up to 360 teraFLOPS, using 8,192 PowerPC CPUs, each running at 700 MHz [25] and arranged in a torus interconnection network [26]. Nevertheless, the scalability of the HPC approach comes at the expense of an extremely high power consumption budget (usually, in the order of hundreds of kilowatts) and a low level of parallelism that restricts real-time execution of large-scale networks.

2.2 Full-Custom Design Approaches

In general, HPCs are too slow to execute large simulations of SNNs and do not scale efficiently to support future networks [5]. Thus, to overcome the scalability problem, alternatives approaches such as full-custom hardware implementations have been explored.

The Neurogrid project [27] built a mixed-signal desktop *supercomputer* for neuroscientists. It uses analog computation to emulate ion-channel activity and a digital communication scheme to support synaptic connections. The main building block is the *neurocore*, which is able to accommodate a total of 65,536 quadratic integrate-and-fire neuron models, and it uses an external FPGA and bank of SRAMs for digital communication between neighboring *neurocores*. Even though Neurogrid does show good power consumption of 5 W for emulation of 1M neurons using a grid of 16×16 *neurocores* [27], the Neurogrid platform does not allow the software flexibility shown by the HPCs. It also has a limitation on the maximum number of neurons per layer (up to 2,175 neurons) that makes it unable to offer biological real time, due to the firing rate for worst-case scenario drops out of up to 16.9 spikes per second (i.e., ~ 59.9 ms interspike interval) [28].

The FACETS projects [29] proposed a mixed-signal, high density hardware neural network architecture based on a combination of analog neurons and a digital multi-layer bus communication scheme; all of them placed on an uncut wafer. The main processing building block of FACETS is the high input count analog neural network chip (HICANN), which contains up to 512 adaptive exponential integrate-and-fire neuron models (AdEx) [30]. A full wafer can comprise 384 HICANN chips, resulting in a total of 196,608 neurons per wafer. To support the neuron interconnection, this platform uses a combination of hierarchical buses for handling neuron communication inside the wafer, and off-wafer routers implemented on a FPGA for providing the wafer-to-wafer interconnection fabric based on a 2D-torus topology. As a distinctive feature, FACETS is able to offer hardware acceleration with up to 10 μs interspike interval per wafer; however, this comes at the expense of an estimated power consumption of 1 kW per wafer [31]. There is a recent follow-on project called the BrainScaleS [32]; however, no scientific literature is available to date.

SpiNNaker [33] is an ongoing project that proposes a fully digital multiprocessor architecture for executing spiking neural network applications. SpiNNaker aims to compute in biological real time ($\sim$1 ms interspike interval), an estimated of 10^9 neurons [34]. Its main building block is conceived as a node, which incorporates 18 ARM968 processor cores (each of them emulating up to 1,000 neurons) and two NoC routers, one for handling the communication between the microprocessors and the peripherals, and the second for handling the communication between processors and neighbor nodes. The interconnection between each node is handled by an NoC using six links, which is wrapped into a triangular lattice; this lattice is then folded onto a surface of a toroid. However, only 16 out of 18 processors are used for emulating neurons, as one ARM968 is used as a monitor processor, and the remaining processor is kept reserved as a spare for fault-tolerance purposes [35]. Therefore, each node is able to offer up to 16,000 Izhikevich neuron models, having a power consumption budget of 1 W.

SyNAPSE [36] is also an ongoing project that aims to realize a full-custom neuromorphic hardware platform able to scale up to biological levels. Its main building block is a digital, configurable *neurochip* that comprises 256 integrate-and-fire neurons. These neurons can be connected in all-to-all fashion using a crossbar [37]. However, due to the initial stage of the project, some basic design specifications are still unclear, for example: How this single *neurochip* will be interconnected with other chips to realize a large network, or what is the power consumption for a single chip just to mention but a few.

In comparison with Neurogrid and SpiNNaker, FACETS is the only architecture able to show acceleration on hardware beyond the biological real time; however, this comes at the expense of having the highest power consumption ($\sim$1 kW). On the other hand, SpiNNaker shows a power consumption higher than Neurogrid; however, SpiNNaker is able to offer biological real time for a million neurons.

The H-NoC architecture proposed in this paper provides (discussed in the following sections) a good tradeoff between scalability and power consumption. It offers not only biological real time but also hardware acceleration similar to FACETS but with a significant reduction in the total power consumption. The H-NoC architecture is part of EMBRACE [13], [3], a mixed-signal approach to NoC-based-embedded neural information hardware. EMBRACE aims to enable future investigation of embedded neural network applications. In this regard, the authors believe that rather than compete, EMBRACE can be seen as complementary tool for neurocomputers such as SpiNNaker or Neurogrid, due to its possibility to offer "faster than real-time" execution of neural network applications, while keeping a low power budget.

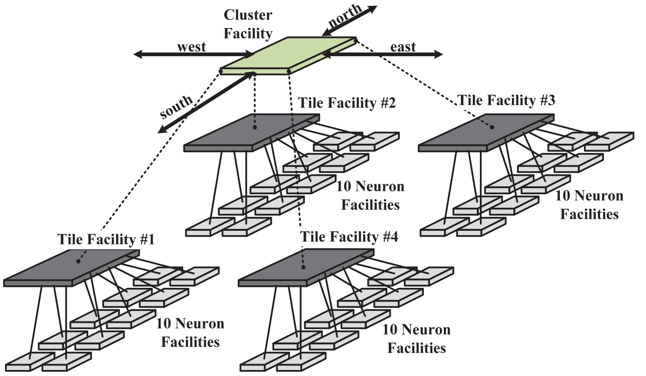


Fig. 2. Proposed scalable H-NoC architecture for implementing clusters of neurons.

3 H-NoC Architecture for SNNs

This section presents an overview of the H-NoC architecture, focusing on the basic concepts and the rationale behind the specifications and design of the H-NoC architecture. Details regarding its implementation can be found in the supplementary material, available online, of this paper.

The proposed H-NoC architecture uses an analog, low power leaky integrate-and-fire neural cell on hardware to model spiking neurons [38]. Nevertheless, the performance and scalability of the proposed H-NoC architecture are analyzed independently of its neuron model's nature (i.e., either analog or digital). The latter is supported on the assumption that each spike event that is absorbed by the routers has been previously digitized, so that spike events can be seen by the NoC routers as incoming digital pulses.

The internal structure of the H-NoC architecture is shown in Fig. 2, and it is based on three modules called 1) *neuron facility*, 2) *tile facility*, and 3) *cluster facility*. Additionally, the on-chip communication of this hierarchical architecture is based on a specialized array of configurable NoC routers, combined with traffic compression and congestion mechanisms to support the local (*intracluster*) and global (*intercluster*) neuron connectivity, as shown in Fig. 3.

This hierarchical approach exploits locality between neurons [16], [17], by allocating groups of 10 neural cells into the so-called *neuron facility*, which is located at the bottom of the hierarchy (see Fig. 2). Spike events generated by neural cells located in these *neuron facilities* are packetized and then issued in the NoC and received by targeted neural cells for processing. The 10 neurons cells per *neuron facility* is a density value to show the functionality of the proposed approach, and it was chosen as different sizes of neural networks are often defined in quantities to the power of ten.

The so-called *neuron facilities* are then connected using a two-level star topology (as shown in Fig. 3a), i.e., the *tile facility* for local neuron connectivity, and the *cluster facility* for local and global neuron connectivity, respectively. As a result, a group of 10 *neuron facilities* (each of them containing 10 neural cells) is gathered in a single *tile facility*. Similarly, a group of four *tile facilities* is then placed within a *cluster facility*, giving a total of 400 neural cells.

If more neural cells were needed, the *cluster facility* can be easily replicated by forming a grid of clusters using a mesh topology. Hence, a mesh formed by a number of *cluster facilities*, "C," can be realized, allowing to scale the total neural cells in the H-NoC architecture by a factor of "C," as shown in Fig. 3b.

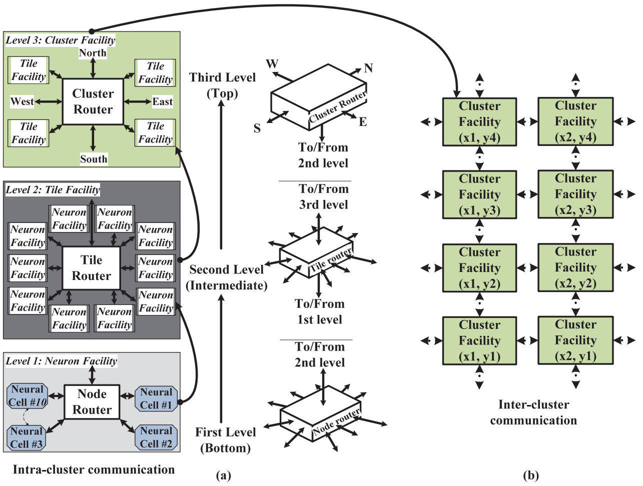


Fig. 3. (a) The internal basic components, and their interconnections, for the *neuron*, *tile*, and *cluster facilities*. (b) Example 2 × 4 cluster mesh.

The scalability of the H-NoC architecture is driven by two key factors, i.e., area/power footprint and throughput. In terms of the area/power footprint, the proposed H-NoC architecture shows a small overhead and a regular mesh layout (see Fig. 3b) that makes it easier to be implemented using traditional CMOS technology. Additionally, in terms of throughput, the proposed H-NoC shows a fixed path delay for both intra- and intercluster communication. Therefore, contrary to what happens on a bus-shared scheme, where adding neurons decreases the communication capacity of the system, in the proposed H-NoC architecture the fixed path delay shown by the *cluster facility* allows the scale up of the whole system, because adding more *cluster facilities* does not increase the cycle time of intercluster communication (see Section 5, for analytical results).

The on-chip communication for a single *cluster facility* is supported by an array of NoC routers of different capabilities, which are distributed as follows: 40 node routers at the bottom of the hierarchy, i.e., one node router per *neuron facility*. Four tile routers in the mid level of the hierarchy, i.e., one tile router per *tile facility*, and one *cluster router* at the top of the hierarchy. Furthermore, each node router provides a fine-grained neuron interconnectivity so that each neuron attached to it can be either configured independently or several neurons can be configured as a group of neural cells to create several neural layers. For example, the *cluster facility* can be configured with 40 neural layers of 10 neural cells each or as a single neural layer of up to 400 neural cells.

The remainder of this section gives a description of the aforementioned *neuron*, *tile*, and *cluster facilities*, and also discusses the different NoC routers and compression technique that allow the on-chip communication between these facilities within the proposed H-NoC architecture. For details regarding the hardware implementation of these facilities, refer to the supplementary material, available online.

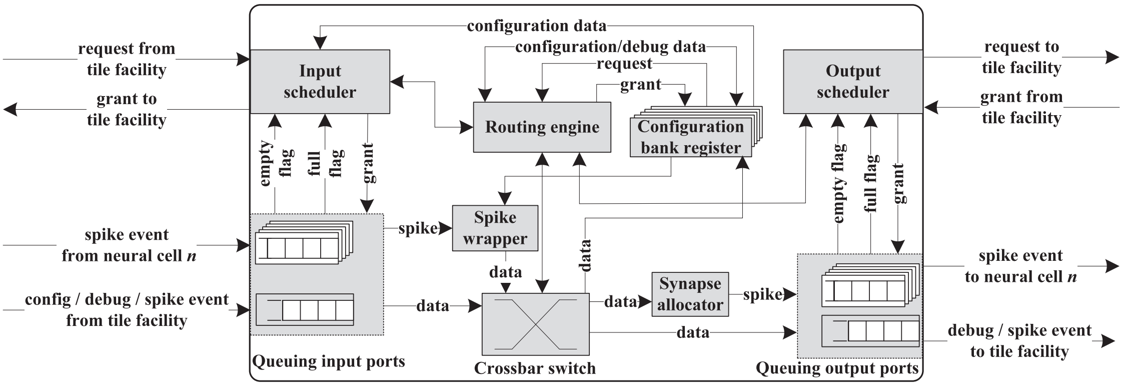


Fig. 4. Node NoC router block diagram.

3.1 Neuron Facility

The first level of the H-NoC architecture is the *neuron facility*, which is found at the bottom of the hierarchy. It connects 10 spiking neural cells ($n = 10$) using the so-called *node router* that converts the incoming data from the neural cells into NoC data packets. The first level accommodates a total of 40 *neuron facilities* ($m = 40$). Therefore, a total of 400 neural cells (i.e., $m \times n$) are comprised at the bottom of the H-NoC architecture.

The *node router* provides the communication infrastructure and incorporates the necessary specialized hardware modules to transfer and receive spike events to/from any of the 10 neural cells allocated in each *neuron facility*. This node router also works as a gateway to reach other neurons located either inside the same or in other neighboring *cluster facilities*. To do this, all the information regarding the type of neuron connectivity, number of neural cells per layer, enable of spike compression technique, and so on, are stored in the *configuration bank registers*, which can be found in each *node router*.

The *configuration bank registers* can be reconfigured at any time, but more importantly, their information are used to assemble the NoC packets and also to support the routing decision making in each router. The routing algorithm for the *node router* follows a distributed-base implementation, i.e., each router computes the next routing address based on the header information and target address contained on the NoC data packet. This distributed approach avoids the use of large lookup tables and replace them by using the combination of compact combinational logic along with the information retrieved from the *configuration bank registers* to route all the data packets, following a uni/multi/broadcast fashion.

Due to all the information of the target neurons are encoded on the NoC packets, upon arrival of one packet, each node router first needs to validate the type of on-chip communication by checking its header information, and second performs a simple comparison between the target address contained on the NoC packet and the local *node router* address. If there is a match between them, the NoC data packet is absorbed by the router, if not the packet is either discarded or bypassed to another router. A detailed block description of the *node router* is shown in Fig. 4, while the packet layout of the aforementioned *bank configuration registers* is depicted and explained in detail in the supplementary material, available online.

The main functionalities of the *node router* are summarized as follows: The router engine is the core and is implemented as a synchronous FSM that can work in three different modes namely, configuration, execution, and debug. During the configuration/debug mode, packets follow a top-down data flow, i.e., the router located at the top of the hierarchy receives the data packets that are coming from the programmer.

During the execution mode, the routing engine coordinates the data transfer that is generated either by the neural cells or from other *tile/cluster facilities*. Once the *node router* receives a spike event from a neural cell, a packetization process is initiated by the *spike wrapper* module. The *spike wrapper* module is located after the input ports so that single or multiple spike events can be converted into a valid NoC data packet, which consists of 48-bits or 56-bits, depending of its type (i.e., execution mode and extended mode; the latter is used to configure neural cells from neighboring *cluster facilities*, when more than 400 neurons needs to be placed in a single neural layer). The packet is organized in three fields: header, source address, and target address. The *spike wrapper* uses the information contained in the *configuration bank registers* to append the correct values for the three aforementioned fields.

When a valid NoC data packet representing a spike event has arrived to its final destination, a spike allocation process takes place and is done by the *synapse allocator* module that is situated at the output of the crossbar. The *spike allocation* process consists of calculating the index value for the synapse position that is going to receive the spike event within the neural cell. To do this, the source address field contained in the NoC data packet is used as an input argument for the *synapse allocator* to compute the synapse index. Implementation details about the synapse index computation are given in the supplementary material, available online.

3.2 Spike Compression Technique

The *node router* also provides a spike compression technique, which exploits the low-firing rate (up to 1-ms interspike interval) shown by biological neurons [1], [4], thus reducing traffic overhead and improving throughput on the network, and to the best of authors' knowledge, this is the first time this technique is explored for NoC-based SNN hardware implementations.

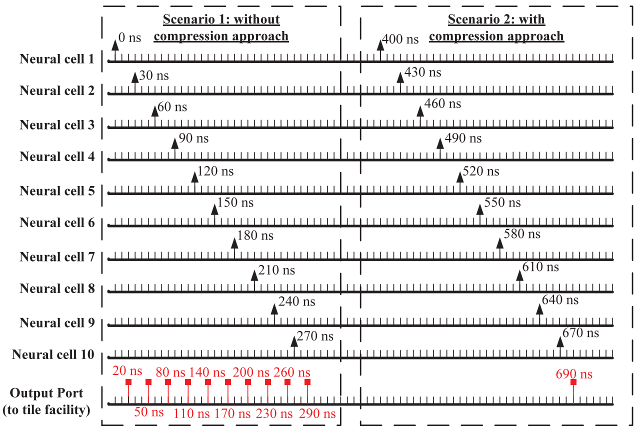


Fig. 5. Node router performance with/without spike compression.

The motivation for implementing this compression technique is founded on the fact that large numbers of neurons fire in irregular bursts [6], causing significant injection of traffic in the NoC. Additionally, not all neural cells in the complete network are expected to generate spikes at the same time [1], however significant numbers of them can. These irregular traffic patterns can have a major impact on the packet latency and can ultimately lead to traffic congestion. Consequently, depending on the number of incoming requests, the input scheduler ignores idle ports and focuses its attention on those that are active and handling spike events. The latter provides the opportunity for spike event compression, where spike events generated by different neural cells within the same *neuron facility* are compressed into a single packet. The compression spike event is achieved by using a *bitmask* approach to annotate spike events generated by neural cells within a predefined time window, decreasing spike traffic volume without affecting the network performance.

The time window range used in this technique expands from 10 ns to 100μs. Large time windows are sufficient to match the integration time of biological spiking neuron models [4], i.e., time for the model to internally compute the presynaptic events that is approximately over five times faster (i.e., $\sim$200 μs) than that of the firing time. A small time window can also be used when the proposed H-NoC needs to work on an acceleration regime, where neuron models can be accelerated to generate spike events every 100 ns. In this regard, Fig. 5 shows two example scenarios at the *node router* with and without the compression technique For both scenarios, it is assumed that the *node router* frequency is 100 MHz.

In the first scenario, the spike traffic compression is not enabled. Thus, every spike event generated from the neurons is transferred as an individual packet to the *tile facility*. In the second scenario, the spike traffic compression is enabled to work using a time window of 300 ns. Hence, all spike events generated within that period of time (e.g., from 370 to 670 ns) are compressed into the same NoC data packet. This reduces the amount of traffic issued to the H-NoC fabric. For example, the number of packets issued inside a single *neuron facility* is reduced from 10 to 1. It is important to highlight that this spike compression technique is only valid when the neural cells generating the spike events have the same destination. This should not be seen as a drawback, but more as an opportunity to reduce the traffic overhead, because when the neural cells are arranged in neural layers (e.g., feedforward), all neural cells that correspond to the same neural layer are generating spikes that eventually will reach the following layer; therefore, all neural cells share the same final destination.

3.3 Tile Facility

The *tile facility* is the second level of the hierarchy and uses the so-called *tile router* in a star topology to group 10 *neuron facilities*. The proposed H-NoC architecture comprises a total of four *tile facilities* as shown in Fig. 2, where each *tile* contains 10 *neuron facilities* (i.e., 100 neural cells). Similar to the *node router*, each *tile router* supports unicast, multicast, and broadcast communication protocols; this combined with the star topology used to interconnect the 10 *neuron facilities* provides an efficient way to manage the traffic demand of the neurons allocated in the mid level of the H-NoC architecture. Additionally, the tile router plays a key role within the proposed hierarchical architecture as it receives and manages all data packets coming either from the bottom or the top of the H-NoC architecture.

The *tile router* comprises the following six components, and its internal architecture is shown in Fig. 6: 10 internal input/output ports, an external input/output port, an input/output scheduler, a *tile address register*, a routing engine, and two crossbar switches. The 10 internal input/output ports as well as the external input/output ports provide the physical interconnection link to communicate

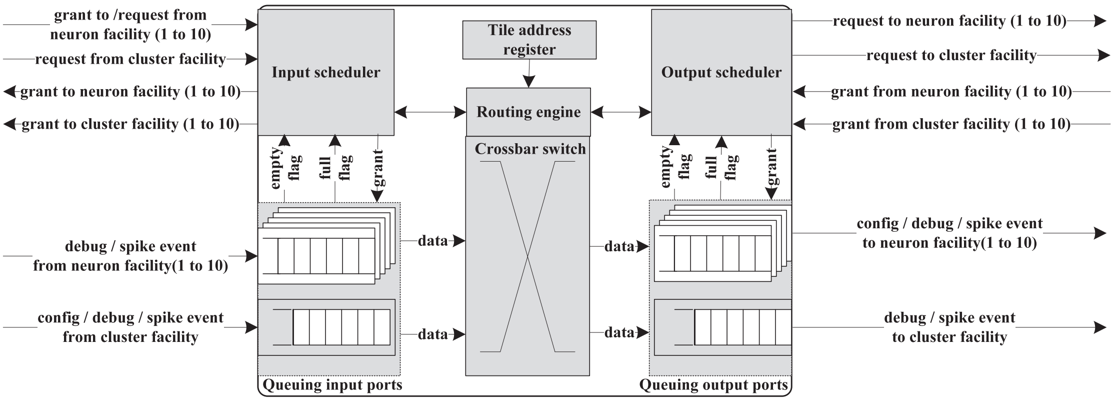

grant to /request from neuron facility (1 to 10)
request from cluster facility
grant to neuron facility (1 to 10)
grant to cluster facility (1 to 10)
Input scheduler
Tile address register
Routing engine
Crossbar switch
Output scheduler
request to neuron facility (1 to 10)
request to cluster facility
grant from neuron facility (1 to 10)
grant from cluster facility (1 to 10)
empty flag
full flag
grant
empty flag
full flag
grant
debug / spike event from neuron facility(1 to 10)
config / debug / spike event from cluster facility
data
data
data
data
config / debug / spike event to neuron facility(1 to 10)
debug / spike event to cluster facility
Queuing input ports
Queuing output ports

Fig. 6. Tile NoC router block diagram.

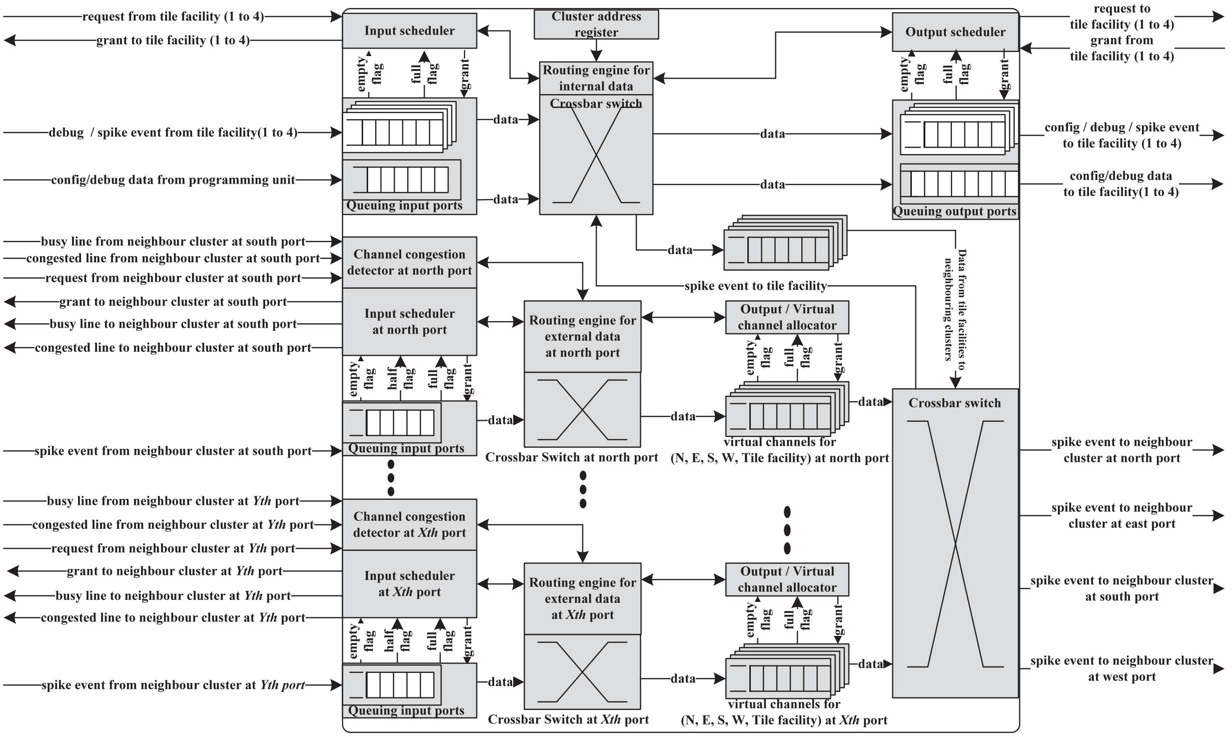

request from tile facility (1 to 4)
grant to tile facility (1 to 4)
Input scheduler
Cluster address register
Routing engine for internal data
Crossbar switch
Output scheduler
request to tile facility (1 to 4)
grant from tile facility (1 to 4)
empty flag
full flag
grant
data
debug / spike event from tile facility(1 to 4)
config/debug data from programming unit
Queuing input ports
config / debug / spike event to tile facility (1 to 4)
config/debug data to tile facility(1 to 4)
Queuing output ports
busy line from neighbour cluster at south port
congested line from neighbour cluster at south port
request from neighbour cluster at south port
grant to neighbour cluster at south port
busy line to neighbour cluster at south port
congested line to neighbour cluster at south port
Channel congestion detector at north port
Input scheduler at north port
Routing engine for external data at north port
Output / Virtual channel allocator
spike event to tile facility
Data from tile facilities to neighbouring clusters
half flag
spike event from neighbour cluster at south port
Crossbar Switch at north port
virtual channels for (N, E, S, W, Tile facility) at north port
Crossbar switch
spike event to neighbour cluster at north port
spike event to neighbour cluster at east port
spike event to neighbour cluster at south port
spike event to neighbour cluster at west port
busy line from neighbour cluster at Yth port
congested line from neighbour cluster at Yth port
request from neighbour cluster at Yth port
grant to neighbour cluster at Yth port
busy line to neighbour cluster at Yth port
congested line to neighbour cluster at Yth port
Channel congestion detector at Xth port
Input scheduler at Xth port
Routing engine for external data at Xth port
spike event from neighbour cluster at Yth port
Crossbar Switch at Xth port
virtual channels for (N, E, S, W, Tile facility) at Xth port

Fig. 7. Cluster NoC router block diagram.

the *tile facility* with the *neuron facility* (bottom level) and the *cluster facilities* (top level), respectively.

The *tile router* comprises two crossbar switches. The downstream data path receives NoC data packets coming from the *cluster facility*. These NoC data packets are then sent to individual or multiple *neuron facilities* at the same time, depending on the specific target location and also on the type of on-chip communication protocol already specified on the header of the data packet (i.e., uni/multi/broadcast). On the other hand, the upstream data path receives NoC data packets from the *neuron facilities*, where NoC data packets are either sent to the upper level within the proposed hierarchy, i.e., the *cluster facility*, or the NoC data packets can be deviated off the upstream path to the downstream data path, so that any of the 10 *neuron facilities* gathered within the same *tile facility* can absorb the NoC data packets. The size of the upstream crossbars is 10×11, whereas the size for the downstream crossbar 1×10. The reason for the difference in sizes of switch is because the *tile NoC router* needs to provide the physical links for any of the 10 *neuron facility* allocated within a single *tile facility*, to communicate with either, any of the other nine *neuron facilities*, with the *cluster facility* or with itself. The last case is of vital importance as this feedback loop allows to support recurrent neural network connectivity and is discussed in the supplementary material, available online, of this paper. Therefore, more outputs were necessary. An example of different neural network topologies mapped to the proposed H-NoC can also be found in the supplementary material, available online, of this paper.

3.4 Cluster Facility

Finally, the *cluster facility* is located at the top level of the hierarchy, and it groups four *tile facilities* using an adaptive router referred to as the *cluster router*. The *cluster router* allows the exchange of spike events between any of the 400 neurons inside the *cluster facility* and neurons located in neighboring *cluster facilities*. The *cluster router* incorporates an adaptive routing scheme [39], [40], which facilitates router adaptation according to the spike traffic loads, improving router throughput according to the traffic behavior presented across the network. It also supports broadcast and multicast communication with any *cluster facility*, for example, in terms of its four neighbors a neuron fan out of up to 1,600 neuron connections can be achieved. Fig. 7 shows a detailed block diagram of the proposed router, highlighting its main components.

The *cluster router* comprises four internal input and output ports (implemented as two separated internal data paths) to interface with the four *tile facilities*. Another four external input and output ports (also implemented as two separated internal data paths) are also provided to interconnect the *cluster facility* with its four neighbor *cluster facilities* on the north (N), east (E), south (S), and west (W) cardinal directions. Due to the *microarchitecture* of the *cluster router* with four distinct data paths, i.e., internals/external, and up/downstream data paths, the *cluster router* is able to offer four distributed adaptive routing units, i.e., one per each external port. This allows up to four simultaneous spike events to be routed when all input ports request a different output port as a final destination, thereby increasing the throughput for the proposed router. Each control unit associated with the four input ports uses an adaptive routing scheme that combines the strong fairness policy of the round-robin arbiter and the priority scheme of a first-come first-serve approach. This hybrid approach handles spike events according to the traffic behavior presented across the network. Details for the implementation of the aforementioned four data paths along with the adaptive

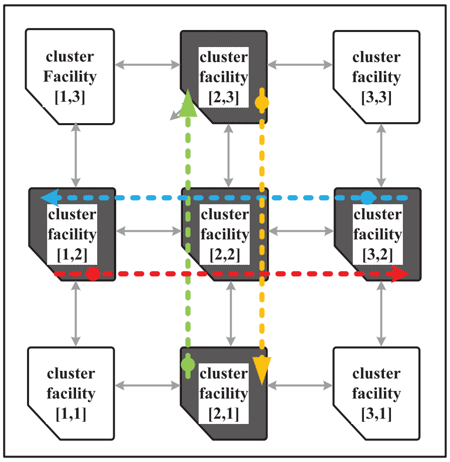


Fig. 8. Evaluation platform of H-NOC using a 3×3 array of clusters.

routing algorithm can be found in the supplementary material, available online, of this paper.

4 Performance Evaluation

This section presents the test bench setup and the traffic load analysis for the proposed H-NoC architecture, where a set of equations to determine the packet delay for the H-NoC are derived. This traffic load analysis allows upper bounds of the H-NoC in terms of throughput capability to be established and also supports the results on scalability via area utilization and power consumption of the proposed H-NoC fabric, for varied SNN traffic loads.

4.1 Testbench Setup

A 3×3 proof-of-concept array of *cluster facilities* of the proposed H-NoC architecture (shown in Fig. 8) has been described in VHDL and evaluated based on simulations. In addition, the 3×3 array of *cluster facilities* has also been implemented and tested on an FPGA to verify its real-time performance on a hardware platform [40]. Moreover, to analyze the power consumption and the area utilization, the VHDL description of a single *cluster facility* was synthesized using Synopsys Design Compiler based on the TSMC 65-nm CMOS technology. A VHDL spike event generator/counter [40] was used in the analysis to inject different traffic loads into the 3×3 array, to emulate incoming spike events (generators) to the clusters and to facilitate the measurement (counters) of packet throughput for this analysis.

All simulations were evaluated using Modelsim, and the following criteria were used in the setup: a NoC data packet width of 48-bits, 100-MHz system frequency, 400 spike generators (SGs)/spike counters (SCs) attached to each cluster, for generating and receiving spike events, and a Poisson neuron firing rate [4], where the interspike interval for generating a new spike event was increased/decreased to emulate different traffic patterns (e.g., fast-spike train, rebound spikes, etc.).

4.2 Traffic Load Analysis

The traffic load analysis of the 3×3 array was performed in four stages, with and without the spike event compression technique to compare its performance advantage. The first stage used two *cluster facilities*, i.e., C[1,2] and C[2,2] connected on the east and west direction, respectively, as shown in Fig. 8. *Cluster facility* C[1,2] was configured to send different spike patterns that were absorbed by the spike counters allocated on C[2,2]. This experiment allowed the measurement of the average packet delay and the average firing rate between two neighboring *cluster facilities* under zero-load traffic.

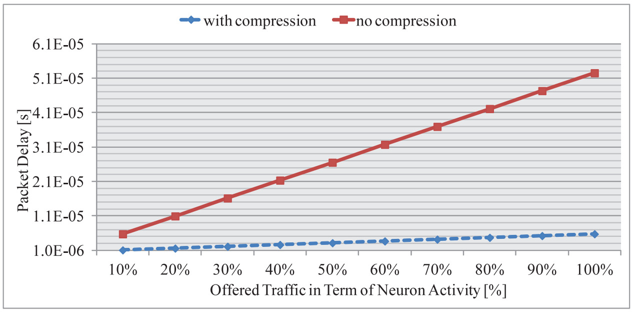


Fig. 9. Average packet delay between two neighbor cluster facilities under different neuron activity traffic levels.

The average packet delay is defined as the average time taken from the moment a spike event is generated until it reaches the allocated synapse associated with the target neuron. The average packet delay gives a measure of how fast neurons can exchange information with each other. The average connection delay can be shown (see (3)) to increase at the same rate as neuron activity increases within the *cluster facility*, as shown in Fig. 9 from simulations. However, due to the spike compression technique, each *node router* is able to accommodate up to 10 spike events in a single NoC data packet under high spike activity periods. Considering a full-load scenario when 400 neurons generate a spike within the same time window, i.e., 100 percent traffic activity on the *cluster facility*, the resulting average packet delay using the compression technique can be significantly reduced from 52.53 to 5.73 μs, hence offering an improvement of up to $\sim 9\times$ compared to a scenario when no compression is used, as shown in Fig. 9.

Similarly, Fig. 10 shows the improvement on the neuron firing rate when the compression approach is used. Typical biological spiking neurons show a firing rate around 100 Hz, but some others can show a firing rate up to 1 kHz [1]. From a hardware point of view, if higher firing frequencies can be achieved, the platform can be used as a neural network hardware accelerator. In this regard, a single *cluster facility* of the proposed hierarchical NoC architecture is able to accommodate high firing frequencies, i.e., between 19 kHz (no compression) and 174 kHz (with compression).

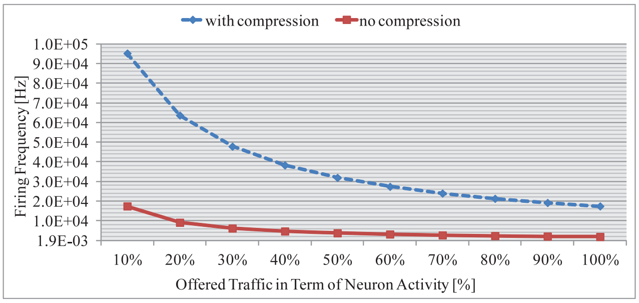


Fig. 10. Average firing rate for a single cluster of neurons.

4.3 Analysis of Packet Delay

The second stage of the traffic load analysis used three *cluster facilities*, i.e., C[1,2], C[2,2], and C[3,2]. The C[1,2] was configured to send spike events to C[3,2], forcing all the spike events to travel through C[2,2]. The aim of this experiment was to measure the time it takes for the spike events to bypass an intermediate *cluster facility*. The result of this experiment together with the results obtained from the first stage was used to derive an analytical equation to model the packet propagation delay (δ) between two clusters. The delay path is composed of three elements namely, upstream delay (node-to-cluster), downstream delay (cluster-to-node), and delay between cluster hops. The upstream delay (T_{up}) represents the packet delay from when packets are generated by the neural cells until the time they all leave the *cluster facility* and can be expressed by

$$T_{up} = \tau_u + \lfloor t_u \times (n_{pg} - 1) \rfloor, \tag{1}$$

where τ_u is a constant that represents the time for the first data packet to appear outside the *cluster facility* (24 clock cycles). Additionally, t_u represents the time interval between packets, i.e., equivalent to pipeline latency (12 clock cycles). Finally, n_{pg} represents the number of packets generated within the *cluster facility*.

Similarly, the downstream delay (T_{down}) represents the delay of data packets from when they go inside the *cluster facility* until they are absorbed by any of the neurons located at the bottom of the hierarchy. The downstream delay is expressed by (2) where τ_d represents the time for the first data packet to be absorbed by any of the neurons within the *cluster facility* (30 clock cycles), t_d represents the time interval between packets, i.e., equivalent to pipeline latency (one clock cycle), and n_{pa} represents the total number of packets absorbed within the *cluster facility*

$$T_{down} = \tau_d + \lfloor t_d \times (n_{pa} - 1) \rfloor. \tag{2}$$

Using (1) and (2), the total packet propagation delay (δ) for a sequence of N_c packets, where $N_c = n_{pg} = n_{pa}$, can be re-expressed as (3). The delay between hops t_h represents the time taken by a packet to bypass a *cluster facility*; this time is a constant and expressed in terms of clock cycles, $t_h = 12$. Additionally, N_h represents the total number of hops or *cluster facilities* that a packet needs to go through to reach its target *cluster facility* and can vary depending on the source to target path

$$\delta = T_{up} + T_{down} + (N_h \times t_h). \tag{3}$$

4.4 Summary of Experimental Results and Large-Scale Analysis

This section summarizes the experimental results on large-scale analysis in terms of throughput, area, and power consumption for the proposed H-NoC architecture. For details of the experimental data and figures, please refer to the online supplementary material, available online, of this paper:

- The proposed H-NoC architecture shows fixed delay when spike events (represented by NoC data packets) are sent between neighboring *cluster facilities*. Results show that the delay for a single spike event between two neighboring *cluster facilities* is 660 ns, while the delay for transferring 400 spike events between them (i.e., 100 percent traffic load for a single *cluster facility*) takes 52.53 μs with no compression, and 5.73 μs when compression is used.
- With an all-to-all connectivity, a single *cluster facility* is able to offer a spike rate equal to 19 kHz (no compression) and 174 kHz (with compression). The latter results show a speed up of 174 $\times$ compared to a biological real-time spike rate (i.e., $\sim$1 kHz). These results were verified on an FPGA and are comparable with [30].
- Results also show a maximum throughput of 3.33×10^9 spikes per second, an area of $0.587\ \mathrm{mm}^2$ and a total power of 13.16 mW, for a single *cluster facility*. This power is significantly lower compared to [30].
- Additionally, five large-scale scenarios (i.e., 10×10, 20×20, 30×30, 40×40, 50×50 *clusters*), using a grid of *cluster facilities*, were analyzed. Results show a linear decay in terms of firing rate; however, the firing rate shown by the neural cells ($\sim$8.7 kHz), for the largest scenario (50×50 *clusters*), is still higher compared to 1 kHz.
- The area utilization and the power consumption per cluster was also analyzed in terms of different number of neural cells within it. The number of neural cells was varied from 400 to 1,000. The results report a promising linear trend for both the area and power, showing a total area of $2\ \mathrm{mm}^2$ and a power consumption of 43 mW, for a single *cluster facility* containing 1,000 neural cells.

5 Conclusion

The authors have proposed a novel hierarchical NoC architecture that enables cluster of neurons to be implemented in hardware using a hierarchical array of NoC routers. The proposed H-NoC architecture is scalable; it offers modular on-chip communication infrastructure for interneuron connectivity, and a spike traffic compression technique to reduce traffic overhead.

The proposed H-NoC architecture was described at the RTL-level implementation and then exhaustively evaluated based on hardware simulations using a wide variety of synthetic spike traffic scenarios. Simulation results demonstrate that the proposed H-NoC approach offers a high, significant throughput of 3.33×10^9 spikes per second. In addition, synthesis results based on 65-nm CMOS technology demonstrates an efficient low-cost area utilization of $0.587\ \mathrm{mm}^2$ and very low power consumption of 13.16 mW for a single *cluster facility* of 400 neurons.

The work presented here is part of a long-term vision to create EMBRACE [13], [3], a mixed signal hardware platform. Results are promising and establish the motivation to continue using the NoC paradigm as a way to overcome the interconnection problems for large-scale hardware SNN implementations. Although having an efficient, high-throughput architecture is important, it is also vital that a balance between increased throughput and minimal power footprint is achieved. Thus, the proposed H-NoC architecture is a step forward in this direction.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their insightful and valuable comments. This work was supported in part by a Vice-Chancellor's Research Scholarship at the University of Ulster.

REFERENCES

[1] W. Gerstner, *Spiking Neuron Models: Single Neurons, Populations, Plasticity,* first ed. Cambridge Univ. Press, 2002.

[2] K.A. Zaghloul and K. Boahen, "A Silicon Retina That Reproduces Signals in the Optic Nerve." *J. Neural Eng.*, vol. 3, no. 4, pp. 257-67, Dec. 2006.

[3] S. Cawley, F. Morgan, B. McGinley, S. Pande, L. McDaid, S. Carrillo, and J. Harkin, "Hardware Spiking Neural Network Prototyping and Application," *Genetic Programming and Evolvable Machines,* vol. 12, pp. 257-280, 2011.

[4] H. Paugam-Moisy and S. Bohte, "Computing with Spiking Neuron Networks," *Handbook of Natural Computing,* G. Rozenberg, T. Back, and J. Kok, eds., pp. 1-47, Springer, 2009.

[5] H. de Garis, C. Shuo, B. Goertzel, and L. Ruiting, "A World Survey of Artificial Brain Projects, Part I: Large-Scale Brain Simulations," *Neurocomputing,* vol. 74, nos. 1-3, pp. 3-29, 2010.

[6] T. Trappenberg, *Fundamentals of Computational Neuroscience,* second ed. Oxford Univ. Press, 2009.

[7] S. Furber and S. Temple, "Neural Systems Engineering," *J. Royal Soc. Interface,* vol. 4, no. 13, pp. 193-206, 2007.

[8] A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Öberg, M. Millberg, and D. Lindqvist, "Network on Chip: An Architecture for Billion Transistor Era," *Proc. IEEE NorChip Conf.*, pp. 166-173, 2000.

[9] L. Benini and G. De Micheli, "Networks on Chips: A New Soc Paradigm," *Computer,* vol. 35, no. 1, pp. 70-78, Jan. 2002.

[10] W. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," *Proc. 38th Ann. Design Automation Conf.*, pp. 684-689, 2001.

[11] T. Theocharides, G. Link, N. Vijaykrishnan, M. Invin, and V. Srikantam, "A Generic Reconfigurable Neural Network Architecture as a Network on Chip," *Proc. IEEE Int'l SOC Conf.,* pp. 191-194, Sept. 2004.

[12] R. Emery, A. Yakovlev, and G. Chester, "Connection-Centric Network for Spiking Neural Networks," *Proc. Third ACM/IEEE Int'l Symp. Networks-on-Chip*, pp. 144-152, May 2009.

[13] J. Harkin, F. Morgan, L. McDaid, S. Hall, B. McGinley, and S. Cawley, "A Reconfigurable and Biologically Inspired Paradigm for Computation Using Network-on-Chip and Spiking Neural Networks," *Int'l J. Reconfigurable Computing,* vol. 2009, pp. 1-13, 2009.

[14] S. Carrillo, J. Harkin, L. McDaid, S. Pande, S. Cawley, B. McGinley, and F. Morgan, "Advancing Interconnect Density for Spiking Neural Network Hardware Implementations Using Traffic-Aware Adaptive Network-on-Chip Routers," *Neural Networks,* vol. 33, pp. 42-57, Sept. 2012.

[15] G. Indiveri, B. Linares-Barranco, T.J. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Häfliger, S. Renaud, J. Schemmel, G. Cauwenberghs, J. Arthur, K. Hynna, F. Folowosele, S. Saighi, T. Serrano-Gotarredona, J. Wijekoon, Y. Wang, and K. Boahen, "Neuromorphic Silicon Neuron Circuits," *Frontiers in Neuroscience,* vol. 5, p. 73, Jan. 2011.

[16] S.H. Strogatz, "Exploring Complex Networks," *Nature,* vol. 410, no. 6825, pp. 268-76, Mar. 2001.

[17] D.J. Watts and S.H. Strogatz, "Collective Dynamics of 'Small-World' Networks," *Nature,* vol. 393, no. 6684, pp. 440-2, June 1998.

[18] D.S. Bassett, D.L. Greenfield, A. Meyer-Lindenberg, D.R. Weinberger, S.W. Moore, and E.T. Bullmore, "Efficient Physical Embedding of Topologically Complex Information Processing Networks in Brains and Computer Circuits." *PLoS Computational Biology,* vol. 6, no. 4, p. e1000748, Apr. 2010.

[19] S. Rodrigo, J. Flich, A. Roca, S. Medardoni, D. Bertozzi, J. Camacho, F. Silla, and J. Duato, "Cost-Efficient on-Chip Routing Implementations for CMP and MPSoC Systems," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 30, no. 4, pp. 534-547, Apr. 2011.

[20] R. Das, S. Eachempati, A.K. Mishra, V. Narayanan, and C.R. Das, "Design and Evaluation of a Hierarchical On-Chip Interconnect for Next-Generation CMPs," *Proc. IEEE 15th Int'l Symp. High Performance Computer Architecture,* pp. 175-186, Feb. 2009.

[21] J.-Y. Kim, J. Park, S. Lee, M. Kim, J. Oh, and H.-J. Yoo, "A 118.4 GB/s Multi-Casting Network-on-Chip with Hierarchical Star-Ring Combined Topology for Real-Time Object Recognition," *IEEE J. Solid-State Circuits,* vol. 45, no. 7, pp. 1399-1409, July 2010.

[22] D. Vainbrand and R. Ginosar, "Scalable Network-On-Chip Architecture for Configurable Neural Networks," *Microprocessors and Microsystems,* vol. 35, no. 2, pp. 152-166, Mar. 2011.

[23] S. Carrillo, J. Harkin, L. McDaid, S. Pande, S. Cawley, B. McGinley, and F. Morgan, "Hierarchical Network-on-Chip and Traffic Compression for Spiking Neural Network Implementations," *Proc. IEEE/ACM Sixth Int'l Symp. Networks-on-Chip,* pp. 83-90, May 2012.

[24] J. Misra and I. Saha, "Artificial Neural Networks in Hardware: A Survey of Two Decades of Progress," *Neurocomputing,* vol. 74, nos. 1-3, pp. 239-255, 2010.

[25] H. Markram, "The Blue Brain Project," *Nature Rev. Neuroscience,* vol. 7, no. 2, pp. 153-159, 2006.

[26] M. Blumrich, D. Chen, P. Coteus, A. Gara, and M. Giampapa, "Design and Analysis of the Bluegene/L Torus Interconnection Network," IBM Research Report RC23025 (W0312-022), vol. 23025, 2003.

[27] "The Neurogrid Website," http://www.stanford.edu/group/brainsinsilicon/neurogrid.html, 2009.

[28] S. Choudhary, S. Sloan, S. Fok, A. Neckar, E. Trautmann, P. Gao, T. Stewart, C. Eliasmith, and K. Boahen, "Silicon Neurons That Compute," *Proc. Int'l Conf. Artificial Neural Networks (ICANN '12),* 2012.

[29] J. Schemmel, J. Fieres, and K. Meier, "Wafer-Scale Integration of Analog Neural Networks," *Proc. IEEE Int'l Joint Conf. Neural Networks,* pp. 431-438, June 2008.

[30] D. Brüderle, M.A. Petrovici, B. Vogginger, M. Ehrlich, T. Pfeil, S. Millner, A. Grübl, K. Wendt, E. Müller, M.-O. Schwartz, D.H. de Oliveira, S. Jeltsch, J. Fieres, M. Schilling, P. Müller, O. Breitwieser, V. Petkov, L. Muller, A.P. Davison, P. Krishnamurthy, J. Kremkow, M. Lundqvist, E. Muller, J. Partzsch, S. Scholze, L. Zühl, C. Mayr, A. Destexhe, M. Diesmann, T.C. Potjans, A. Lansner, R. Schüffny, J. Schemmel, and K. Meier, "A Comprehensive Workflow for General-Purpose Neural Modeling with Highly Configurable Neuromorphic Hardware Systems," *Biological Cybernetics,* vol. 104, nos. 4/5, pp. 263-96, May 2011.

[31] J. Schemmel, J. Fieres, and K. Meier, "Wafer-Scale Integration of Analog Neural Networks," *Proc. IEEE Int'l Joint Conf. Neural Networks,* pp. 431-438, 2008.

[32] "The BrainScaleS Website," http://brainscales.kip.uni-heidelberg.de/, 2011.

[33] "The Spinnaker Project Website," http://apt.cs.man.ac.uk/projects/SpiNNaker/, 2007.

[34] S. Furber and A. Brown, "Biologically-Inspired Massively-Parallel Architectures—Computing beyond a Million Processors," *Proc. Ninth Int'l Conf. Application of Concurrency to System Design,* pp. 3-12, July 2009.

[35] 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,* pp. 1-1, 2012.

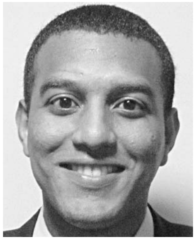
[36] "The SyNAPSE Project Website," http://www.almaden.ibm.com/cs/people/dmodha/, 2011.

[37] J.-s. Seo, B. Brezzo, Y. Liu, B.D. Parker, S.K. Esser, R.K. Montoye, B. Rajendran, J.A. Tierno, L. Chang, D.S. Modha, and D.J. Friedman, "A 45 nm CMOS Neuromorphic Chip with a Scalable Architecture for Learning in Networks of Spiking Neurons," *Proc. IEEE Custom Integrated Circuits Conf. (CICC),* pp. 1-4, Sept. 2011.

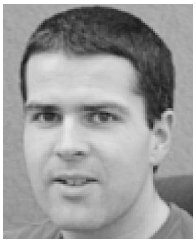
[38] L. McDaid, S. Hall, and P. Kelly, "A Programmable Facilitating Synapse Device," *Proc. IEEE Int'l Joint Conf. Neural Networks,* pp. 1615-1620, 2008.

[39] S. Carrillo, J. Harkin, L. McDaid, S. Pande, and F. Morgan, "An Efficient, High-Throughput Adaptive NoC Router for Large Scale Spiking Neural Network Hardware Implementations," *Proc. Ninth Int'l Conf. Evolvable Systems: From Biology to Hardware,* pp. 133-144, 2010.

[40] S. Carrillo, J. Harkin, L. McDaid, S. Pande, S. Cawley, and F. Morgan, "Adaptive Routing Strategies for Large Scale Spiking Neural Network Hardware Implementations," *Proc. 21th Int'l Conf. Artificial Neural Networks,* pp. 77-84, 2011.



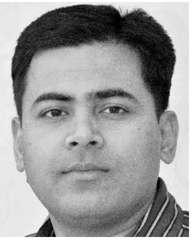
Snaider Carrillo received the BEng (Hons) degree in electronic from the Universidad del Norte, Barranquilla, Colombia, in 2006 and the MSc degree in electronic engineering (With a Magna Cum Laude Distinction) from the Pontificia Universidad Javeriana, Bogota, Colombia, in 2008. He was a research assistant in the Department of Electronics Engineering, Pontificia Universidad Javeriana, Bogota, Colombia, from July 2006 to June 2008. During the summer of 2008, he was a visiting researcher in the Department of Electrical and Computer Engineering, University of Delaware, Newark. Afterward, he was appointed lecturer in the Department of Electronics and Telecommunication Engineering, Universidad Autónoma del Caribe, Barranquilla, Colombia. In September 2009, he received the Vice-Chancellor's Research Scholarship for working toward the PhD degree in computer science at the University of Ulster. He is developing his research project at the Intelligent Systems Research Centre, where he joined the Nanoelectronics Research Team to work on dedicated hardware architectures for large-scale spiking neural network implementations. His research interests relate to: system and network-on-chip (SoC and NoC), embedded systems, FPGAs, GPGPUs, and computer arithmetic. He was the corecipient of the Best Paper Award in the Ninth International Conferences on Evolvable Systems (2010) and also the corecipient of the ENNS Student Grant Award in the 21st International Conference on Artificial Neural Networks (2011). He is also part of the IEEE CIS GOLD Sub-Committee (2010/2011/2012) and the GOLD Rep of the UKRI CIS Chapter (2011/2012). He is a member of the IEEE.



Jim Harkin received the BTech degree in electronic engineering, the MSc degree in electronics and signal processing, and the PhD degree from the University of Ulster, Derry, Northern Ireland, United Kingdom, in 1996, 1997, and 2001, respectively. He is a lecturer at the School of Computing and Intelligent Systems, University of Ulster, Derry. He is a member of the UK Neuroinformatics Node Special Interest Group on Neurally-inspired Engineering and was co-guest editor of a Special Topic in Frontiers of Neuroscience. He has published more than 70 articles in peer-reviewed journals and conferences. His research focuses on the design of intelligent embedded systems to support self-repairing capabilities and the hardware/software implementations of spiking neural networks. He is a member of the Institution of Engineering and Technology and received the Distinguished Learning Support Fellowship in 2006 from the university in recognition of his contributions to teaching.

Liam J. McDaid received the BEng (Hons.) degree in electrical and electronics engineering from the University of Liverpool, United Kingdom, in 1985, where he also received the PhD degree in solid state devices. He is currently a reader at the School of Computing and Intelligent Systems, University of Ulster, N. Ireland. He is currently a guest editor for a special topic entitled "Biophysically Based Computational Models of Astrocyte - Neuron Coupling and Their Functional Significance" to appear in *Frontiers in Neuroscience* and he has coauthored more than 100 publications in his career to date. He is a founder member of the Nanoelectronics Research Group within the Intelligent Systems Research Centre at the Magee Campus of the University of Ulster. His main research interest includes software/hardware implementations of neural-based computational systems, and he has several research grants in this domain. His ultimate vision is to understand and model the mechanisms that underpin self-repair in the human brain thus providing the blue print for advanced architectures that exhibit a fault-tolerant capability well beyond existing computational systems.

Fearghal Morgan received the BSc (Hons.) degree in electrical and electronic engineering and the PhD degree from Queens University, Belfast, United Kingdom, in 1981 and 1986, respectively. His PhD research investigated buried channel MOSFET design, fabrication, and characterization. He is the director of the Bio-Inspired and Reconfigurable Computing Research Group, Electrical and Electronic Engineering, National University of Ireland (NUI), Galway, Ireland. He was a senior design engineer of network and router products with Digital Equipment Corporation, Ireland. He has been with NUI Galway, since 1996, following four years with the Institute of Technology Tallaght, Dublin, Ireland. He received the NUI Galway President's Award for Teaching Excellence in 2009. His research interests include NoC-based embedded hardware spiking neural network architectures and applications, and the integration of online training and assessment material with remote FPGA hardware. He is a fellow of the Institution of Engineers of Ireland.



Sandeep Pande received the BE (Hons.) in electronic engineering in 1996 and the MTech (Hons.) degree in electronic engineering in 2000 from Nagpur University, India. He is currently a researcher at Bio-Inspired Electronics and Reconfigurable Computing Group, Electrical and Electronic Engineering, National University of Ireland, Galway, where he is currently working toward the PhD degree in the design exploration for EMBRACE Hardware Spiking Neural Network Architecture. He was with leading research and semiconductor companies, where he developed and deployed system-level design methodologies and multiprocessor SoC architectures. His research interest includes design and performance enhancement of MP-SoC and NoC architectures and development of nature-inspired computing architectures.

Seamus Cawley received the BE (Hons.) degree in electronic and computer engineering from the National University of Ireland (NUI), Galway, in 2008. Since October 2008, he has been working toward the PhD degree at the Bio-Inspired Electronics and Reconfigurable Computing Research Group, NUI, where he is working on the development of a scalable network-on-chip-based hardware spiking neural network. He is funded by Science Foundation Ireland through the Efficient Embedded Digital Signal Processing Research cluster.

Brian McGinley received the BEng (Hons.) degree in electronic and computer engineering and the PhD degree from the National University of Ireland, Galway, Ireland. His PhD research investigated spiking neural networks and genetic algorithm parameter adaptation. Since 2008, he has been with NUI, as a postdoctoral researcher with the efficient embedded digital signal processing research cluster. His current research interests include heuristic algorithms, embedded systems, reconfigurable hardware, and digital signal processing applications for mobile digital healthcare.