An Efficient, High-Throughput Adaptive NoC Router for Large Scale Spiking Neural Network Hardware Implementations

S Carrillo[1], J Harkin[1], L McDaid[1], S Pande[2] and F Morgan[2]

[1] Intelligent Systems Research Centre (ISRC),
University of Ulster, Magee Campus, Derry, Northern Ireland.
carrillo_lindado-s@email.ulster.ac.uk
[2] Bio-Inspired Electronics and Reconfigurable Computing Research Group (BIRC),
National University of Ireland, NUI Galway, Galway, Ireland.
sandeep.pande@nuigalway.ie

Abstract. Recently, a reconfigurable and biologically inspired paradigm based on network-on-chip (NoC) and spiking neural networks (SNNs) has been proposed as a new method of realising an efficient, robust computing platform. However the use of the NoC as an interconnection fabric for large scale SNN (i.e. beyond a million neurons) demands a good trade-off between scalability, throughput, neuron/synapse ratio and power consumption. In this paper an adaptive NoC router architecture is proposed as a way to minimise network delay across varied traffic loads. The novelty of the proposed adaptive NoC router is twofold; firstly, its adaptive scheduler combines the fairness policy of a round-robin arbiter and a first-come first-served priority scheme to improve SNN spike packet throughput; secondly, its adaptive routing scheme (verified using simulated SNN traffic) allows the selection of different NoC router output ports to avoid traffic congestion. The paper presents the performance and synthesis results of the proposed adaptive NoC router operating within the EMBRACE architecture. Results illustrate that the high-throughput, low area and low power consumption of the adaptive NoC router make it feasible for use in large scale SNN hardware implementations.

Keywords: Adaptive Router Architecture, Spiking Neural Networks, Network-on-Chip, EMBRACE.

1 Introduction

In the last sixty years, several computational models such as the *Hodgkin and Huxley*, the *leaky integrate & fire* and the *Izhikevich* have been proposed to mimic to a certain degree the biological behaviour of real neurons. These computational neuron models have led to the creation of the interesting and powerful bio-inspired Spiking Neural Network (SNN) computational paradigm [1]. The challenge is to develop a complex high-performance synapse/neuron interconnection pattern, implemented in an electronic device that exhibits low power consumption, reconfigurable capabilities, intrinsic parallelism and a high level of scalability.

The complexity of inter-neuron connectivity is prohibiting the progress in hardware toward biological-scale SNNs as the rapid increase in the ratio of fixed connections to the number of neurons is limiting the size of the network. To overcome this issue, initial research has focused on the Network-on-Chip (NoC) interconnect paradigm as a possible mechanism to support scalability. Nevertheless, the use of the NoC as an interconnection fabric for large scale SNNs (i.e. beyond a million neurons) demands a good trade-off between scalability, throughput, neuron/synapse ratio and power consumption. Consequently, the router itself plays an important role, mostly because its hardware architecture has a major impact on the following parameters:

- *Power Consumption:* the router is the communication point to which synapses and neurons are attached; this implies that the number of routers increases proportionally with the number of neurons. Hence, the power consumption for large scale SNN hardware implementations increases as the major contributor for this power consumption is the interconnection fabric (i.e. routers, index tables, etc). The neuron model usually has a power consumption approximately six orders of magnitude smaller than that of the interconnection fabric [2].

- *Throughput:* the router is also responsible for managing SNN spike events. However the traffic pattern shown by spiking neurons is highly asynchronous and non-uniform [1]. Hence, an effective arbitration policy is desired. This policy should be dynamically adaptable depending on the traffic behaviour and should also route and deliver as many SNN spike events as possible in a short period of time without affecting the traffic performance and without incurring any significant hardware overhead.

- *Traffic Congestion:* the typical firing rate for a biological neuron is between 10 ms - 30 ms [1]. However, as the number of neurons increases, the increasing number of SNN spike events presents routers with the more difficult task of accomplishing real-time routing without SNN spike packet loss. Therefore, for large scale SNN hardware implementation, routing algorithms which implement traffic congestion management features to reduce traffic congestion impact are required.

In this paper an adaptive NoC router architecture is proposed and its feasibility for large scale EMBRACE [3] SNN hardware implementation is presented.

The rest of this paper is organised as follows: Section 2 presents the motivation for this research and summarises current work regarding NoC-based SNN hardware implementation. Section 3 discusses the proposed adaptive NoC router architecture incorporated within the EMBRACE architecture. Section 4 presents results and analysis of the proposed adaptive NoC router architecture in term of area utilisation, power consumption and spike packet throughput. Finally Section 5 provides a brief discussion regarding large scale SNN hardware implementations and conclusion.

2 Motivation and Previous Works

Traditionally, software approaches are too slow to execute a long simulation of SNNs and do not scale efficiently [4]. Thus, researchers have explored alternative hardware SNN solutions using FPGAs and GPUs that provide a fine-grained parallel architecture and a 2D mesh interconnect topology [5], [6]. The authors highlight [4] that FPGA approaches have several limitations such as inefficient area utilisation and a Manhattan style interconnect. [7] indicates that the problem of accessing memory in a coherent manner and limited memory bandwidth, are major drawbacks for SNNs based on GPU platforms. Moreover, neither the FPGA nor the GPU architectures are power efficient and they have limited on-chip weight storage capabilities.

Therefore, it is necessary to look to custom hardware to provide area and power requirements which can support large scale hardware SNN realisation with dense neuron interconnection. Traditional approaches using a bus-shared topology offer a simple and inexpensive channel to interconnect several neurons. However, in [8] the authors compare the performance of the bus topology with different interconnection networks topologies, and conclude that a bus topology does not scale to allow interconnection of a large number of SNN processing elements (PEs). Furthermore, a bus topology is not able to guarantee real-time execution since the latency of the network increases proportionally to the number of PEs connected to the shared-bus.

In [9] the Network-on-Chip (NoC) interconnect paradigm is introduced as a promising solution to solve the on-chip communication problems experienced in Systems-on-Chip (SoC) computing architectures, where generally high throughput and high interconnect capability is required. In general, NoC architectures are composed of a set of shared PEs, network interfaces, routers and channels, which are arranged in a topology depending on the application. In the context of SNNs, these PEs refer to the neuron models attached to the NoC routers placed throughout the network. Channels are analogous to the synapses/axons of spiking neurons. The SNN topology in this case refers to the way spiking neurons are interconnected across the network.

The concept of using NoCs in SNNs for large scale hardware implementations has been reported in [10], [11], [12] (summarised in Table 1). Although a few of these projects exhibit good throughput and some others provide a high Quality of Service (QoS), their major drawback is that they do not provide an adequate mechanism to deal with traffic congestion with scaling of the SNN size. This is important in achieving efficient large scale SNN hardware implementations.

Table 1. SNN hardware implementation examples using NoCs

Project Reference	Quality of Service (QoS)	Congestion Mechanism	Power [mW]	Throughput [Gbps]
Spinnaker [12]	Best Effort	No	64	14.4
Facets [11]	Best/Guaranteed Effort	No	NA	6.1
Theocharides. et al [10]	Best Effort	No	NA	0.1

3 Adaptive NoC Router

In previous work [3] the authors proposed EMBRACE, a custom field programmable neural network architecture that merges the programmability features of FPGAs and the scalable interconnectivity of NoC router strategy to implement large scale spiking neural networks with a custom low-area/power programmable synapse cell, which has characteristics similar to real biological synapses [13]. The proposed NoC strategy uses individual routers to group *n* synapses and the associated neuron using a novel structure referred to as a neural tile. The neural tile is viewed as a macro-block of EMBRACE and its novelty resides in the merging of analogue synapse/neuron circuitry with NoC digital interconnect to provide a scalable and reconfigurable neural building block. The EMBRACE NoC architecture is a mesh-based two-dimensional array of interconnected SNN neural tiles (each connected in N, E, S, W directions) as illustrated in Fig. 1. Spike exchange between neural tiles is achieved by routing packet-based spike information through NoCs connected to neural tile ports. Moreover, the EMBRACE architecture supports the programmability of SNN topologies on hardware, providing an architecture that enables the accelerated prototyping and hardware in-loop training of SNNs. In this regards, a parallel research project called EMBRACE-FPGA [14], has enabled the development of a 32 neuron, 32-synapses per neuron hardware SNN evolution platform, executing a range of applications, and allowing refinement of EMBRACE architecture selection.

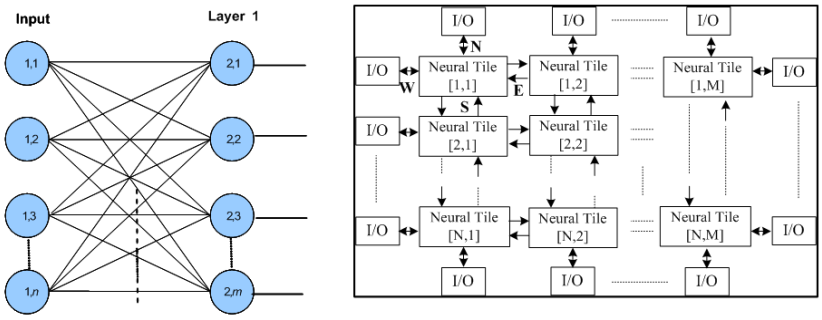

Fig. 1. EMBRACE architecture overview [3]

Although the previous NoC router design [3] exhibits good latency and area performance, i.e. the router can process incoming data packets every 10 clock cycles with source packet generation requiring 12 cycles using 234 LUTs for a Virtex 4 device, its non-adaptive architecture makes it difficult to overcome the congestion problem present in a large scale SNN implementation. Therefore, an adaptive router architecture has been proposed and investigated as a way to route packets of spike events efficiently throughout the network whilst trying to balance congestion in the network (i.e. increase the throughput).

The adaptability of the proposed router can be described in two dimensions:

- An adaptive arbitration policy (AAP) module which combines the fairness policy of a round-robin arbiter and the priority scheme of a first-come first-served (FCFS) approach, enabling improved router throughput according to the traffic behaviour presented across the network.

- An adaptive routing scheme (ARS) module which enables the selection of different router paths to avoid traffic congestion, based on pattern traffic and a channel congestion detector (CCD).

Those modules and their interconnection are explained in more detail in the following sub-sections. Fig. 2 illustrates the proposed adaptive router and its interconnection with the EMBRACE neural cell.

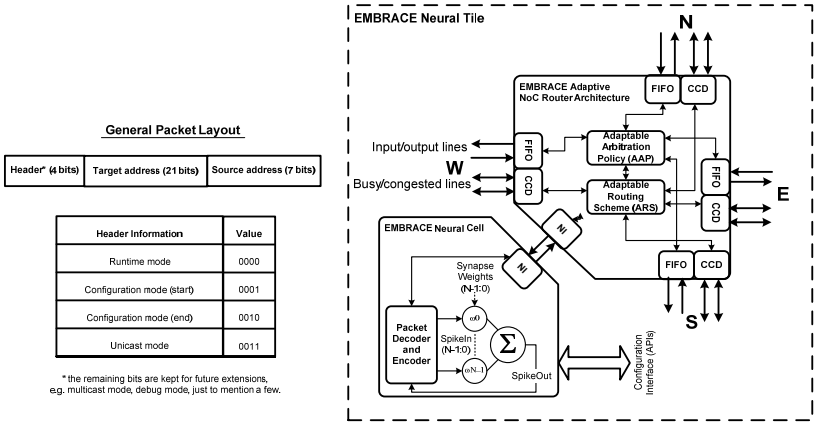

Fig. 2. EMBRACE neural cell and adaptive NoC router structure and connection

3.1 Adaptive Arbitration Policy

A key property of an arbiter is its fairness in the way that it provides equal service to different network traffic requests. Accordingly, several arbitration policies have been proposed [15]. However, almost all of them are either based on strong fairness (i.e. round-robin arbiter) or weak fairness (i.e. FCFS arbiter). A round-robin arbiter exhibits a strong fairness to service each port, because it allocates the lowest priority to the port that has been serviced in the previous round and gives the highest priority to the next port. This approach seems to be good for a heavy load scenario where all router ports are busy, since it gives equal priorities to all of them. However, this strong fairness also introduces two factors that alter the latency in the router, namely, the number of router ports is proportional to the router latency, and the position of the round robin arbiter when there is an incoming spike [16]. On the other hand, a FCFS arbiter gives the highest priority to the first event that occurs. Thus, a FCFS arbiter would be good for those traffic scenarios where only one port is busy (or at most a few ports are busy at the same time) since the router

does not spend extra clock cycles servicing inactive or unused ports. However, contrary to the round robin approach, this weak fairness is not feasible for heavy traffic load scenarios since the probability of discarded packets increases due to the arbiter priority being given to the first port that requests its attention. Consequently, the authors propose an adaptive arbitration policy (AAP) which combines the strong fairness policy of the round-robin arbiter and the priority scheme of a FCFS approach. This hybrid approach improves router throughput according to the traffic behaviour presented across the network. The proposed AAP uses a spike event register to store information regarding any new spike event for each port input buffer. Five distributed control units (i.e. one for each port) allow the scheduler to manage thread communication without incurring task-switching overhead. Therefore only the input buffers that contain information will be serviced, avoiding wasted clock cycles servicing those input buffers that do not contain packets. In the same way, when a heavy load traffic scenario occurs, all of the ports will be serviced based on a round-robin arbitration scheme. The pseudo code used for the proposed arbitration module is illustrated in Fig. 3.

```
1-  entity Adaptable_Arbitration_Policy
2-  {
3-    Input     : InputBufferDataPresent;
4-    Output    : SelectedPort;
5-    Register  : SpikeEventRegister;
6-  }
7-  Begin
8-    Asynchronous_Process1(InputBufferDataPresent!)
9-      SpikeEventRegister <= (ActivePortNumber)
10-
11-   Asynchronous_Process2(SpikeEventRegister)
12-   {
13-     Case (TwoBusyPorts)
14-       task-Switching(SpikeEventRegister.PortNumbers)
15-     Case (ThreeBusyPorts)
16-       task-Switching(SpikeEventRegister.PortNumbers)
17-     Case (FourBusyPorts)
18-       task-Switching(SpikeEventRegister.PortNumbers)
19-     Case (FiveBusyPorts)
20-       task-Switching(RounRobin.PortNumbers)
21-     Default
22-       task-Switching(FirtsComeFirstServe.PortNumber)
23-   }
24-   Output(SelectedPort.PortNumber)
25- End;
```

Fig. 3. Pseudo code for the adaptive arbitration policy (AAP) module

3.2 Adaptive Routing Scheme

The adaptive routing scheme module for the proposed router is composed of three main elements. Firstly, a routing algorithm that is based on an XY routing approach [15] and receives a default output port direction from the AAP module. Secondly, the channel congestion detector (CCD), illustrated in Fig. 4, uses information received from neighbouring routers to generate an alternative output port direction and passes this information to the adaptive routing decision (ARD) module, see Fig. 5. The ARD module takes the default output port direction given by the XY routing algorithm, and based on the information generated by the CCD, the proposed default output port is granted, or the alternative output port direction which is generated according to the traffic information received from the CCD, is selected. The CCD and ARD components are described below.

The Channel Congestion Detector (CCD): Fig. 4 illustrates the CCD module which provides a means of detecting the current state of SNN packet traffic in any given direction. For any given direction the CDD module can detect whether the forward N, E, S or W channels are free, busy or congested, as follows:

a. Free: the input FIFO is empty or less than half-full.
b. Busy: the input FIFO is half-full.
c. Congested: the input FIFO is full.

The CCD module uses a combination of logical two-input AND gates and two-input OR gates. Whenever a router FIFO buffer is full, it asserts the FIFO full signal so that the full status can be detected by the CCD and propagates logic '1' to each of its associated AND, as illustrated in Fig 4. Similarly, if a FIFO is half-full then it will generate logic '1' to its associated OR gate. The AND and OR gates are connected in a daisy chain between each router. The routing decision output signal of each router appears as a two bit value, one belonging to the congested line and the other belonging to the busy line, as shown in Fig. 4. When both lines are '0' then the channel is free. Logic '1' on the busy line indicates that a FIFO can take a packet but that it has a limited capacity. The output of this logic element will be four two-bit outputs, i.e. two bits for each direction. The traffic information (i.e. the congested, busy o free status per channel) generated from the CCD, is then forwarded as an input to the adaptive routing decision (ARD, in order to select the output port direction.

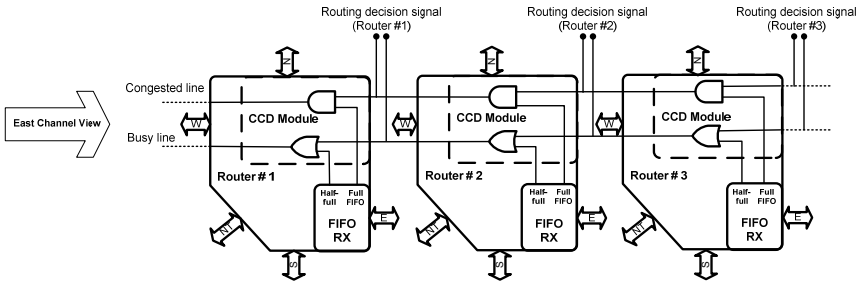

Fig. 4. Channel congestion detector (CCD). This figure only shows the situation when looking along the east channel for clarity purposes. However, the 'look-ahead' facility is replicated in all N, E, S and W directions

The Adaptive Routing Decision (ARD): this module selects the forwarding port direction for routing spike packet data. The ARD module considers the two bits generated by the XY routing algorithm as the priority/default direction and the input from the CCD. The default output port direction given by the XY routing algorithm is used to index the routing table shown in Fig. 5. Those directions are '00', '01', '10' and '11', which correspond to N, E, W and S, respectively. The output of this table is the two alternative adapted directions for the given input. For example, if the input direction is north ('00') then the output of the lookup table is either "01" or "10". These two outputs represent E and W direction, respectively. These values are both used as select lines in two individual multiplexers, which are used to compare these values with the busy lines from the congestion detector to check if either of these channels is busy.

The proposed ARD occupies a small area, is scalable and is a low power element. It therefore adds very little overhead to the NoC router architecture.

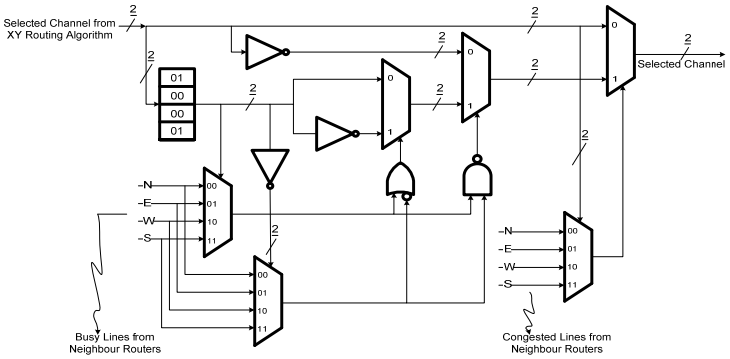

Fig. 5. Adaptive routing decision (ARD) module

4 Performance Analysis

This section presents results on the throughput capability of the proposed adaptive router for varied SNN traffic loads, and benchmarks its performance against existing approaches. The area and power requirements of the proposed additional circuitry are also highlighted.

4.1 Methodology

A VHDL implementation of the proposed adaptive router architecture has been created in order to evaluate its performance. The router is characterised by its packet throughput, area utilisation and power consumption parameters. A SystemC spike event counter/generator testbench facilitated measurement of packet throughput. Area and power metrics have been obtained using the Synopsys Design Compiler tool for TSMC 90nm CMOS technology. The measurement setup was inspired by [15] and verified in [16]. This setup proposed the attachment of terminal instruments such as counters and generators at each router port. The spike event generator includes a packet source module to generate spike data according to the spike packet layout illustrated in Fig. 2. The spike event generator also defines the traffic pattern, packet length and the spike injection rate (i.e. the time between spike events). The spike event counter measures the SNN output spike rate and deduces the spike throughput and the number of unsuccessfully routed (dropped) spike packet. The relationship between the depth of the input FIFOs and their impact on the maximum throughput of the router and the total area power consumption is analysed within the simulation framework the depth of the input FIFOs varies between 1 up to 5. The inter-router packets data width is 32-bits, the width of the spike packet. The router operating frequency is 100MHz and the spike counter sample window time used is 1ms. A pre-count stage is applied before each counter window to allow the router to reach steady state operation.

4.2 Performance Results

Several experiments have been carried out to assess the packet throughput for the proposed adaptive NoC router. These experiments have examined the impact of the spike inject rate (SIR) variation on the average adaptive router packet throughput. Router performance has been compared with that of a non-adaptive router (i.e. a round-robin equivalent) using an input FIFO depth of 5. Fig. 6 illustrates the packet throughput advantages in using the proposed adaptive router strategy. This demonstrates equal throughput performance for both adaptive and non-adaptive router when an SIR of 20 is applied (i.e. a spike packet is generated every 20 clock cycles). However, when an SIR of 2 is applied, the adaptive router achieves almost double the throughput of the round robin-based router. This is a typical traffic scenario for spiking neurons in burst mode. The advantages of the proposed adaptive router approach are as follows:

- When not all router ports are used, the adaptive router skips over idle router ports. Several clock cycles can be saved compared to the round-robin approach and the overall throughput can be increased. Figures 6a and 6b show the result obtained when one and three ports are used.

- When all router ports are busy, a packet throughput advantage using the proposed adaptive router occurs when the SIR value is less or equal to the number of ports minus one (i.e. SIR = 4), due to the non-adaptive router reaching the saturation level, i.e. the router is not able to deliver the packets as fast as they are being generated [15]. Therefore it is impossible for the round-robin arbiter to service all ports adequately since the SIR is smaller than the time available to the router to process the incoming packets. As a result, the unattended ports drop packets and the throughput saturates.

4.3 Evaluation

Table 2 summarises the power consumption and area utilisation for the proposed adaptive NoC router. These parameters have been obtained using the Synopsys Design Compiler tool for the TSMC 90nm CMOS technology. A router clock frequency of 100 MHz has been used and the dynamic power consumption metrics have been obtained based on a fully loaded traffic scenario, where all neurons spike at the same time. Table 2 also shows the trade-off between the depth of the input FIFO and the maximum throughput per router. In addition, Tables 3 and 4 compare the performance of the proposed router with other existing approaches [10], [11], [12]. Table 3 highlights the routing algorithms used for the NoC routers. Table 4 highlights a high throughput of 16Gbps for the proposed adaptive NoC router whilst exhibiting a low power overhead of 1.86mW. The adaptive NoC router achieves a higher throughput performance than existing approaches. The authors are aware that the proposed router does not contain any index table to implement a multicasting scheme as in [12], which would increase the presented area and power metrics for the proposed router. However the throughput would remain the same since the arbitration policy would be independent of a future multicasting approach.

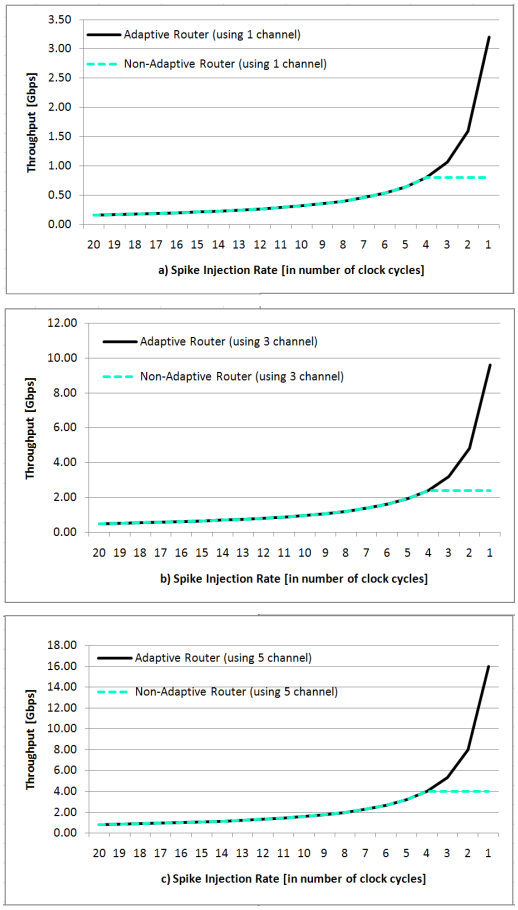

Fig. 6. Relationship between the spike injection rate and the throughput per router, under different traffic load

Table 2. Synthesis summary for the proposed router obtained from Synopsys Design Compiler tool based on the TSMC 90nm CMOS technology library

Input FIFO [Depth]	Dynamic Power [mW]	Leakage Power [mW]	Total Power [mW]	Area [mm^2]	Avg. Throughput [Gbps]
1	0.82	0.13	0.95	0.039	13.44
2	1.04	0.14	1.19	0.041	14.08
3	1.24	0.15	1.39	0.045	14.72
4	1.44	0.17	1.61	0.048	15.36
5	1.67	0.19	1.86	0.054	16.00

Table 3. Comparison of the proposed router against other existing approaches

Project Reference	Neuron Model	NoC Topology	Routing Algorithm
This work	LI&F	2D Mesh	Adaptive XY routing
Spinnaker [12]	Izhikevich	2D Triangular Torus	Node table routing
Facets [11]	LI&F	2D Torus Mesh	iSLIP
Theocharides. et al [10]	I&F	2D Mesh	XY routing

Table 4. Comparison of router performance against other existing approaches

Project Reference	Quality of Service (QoS)	Congestion Mechanism	Throughput [Gbps]	Power [mW]
This work	Best Effort	Yes	16.0	1.86
Spinnaker [12]	Best Effort	No	14.4	64
Facets [11]	Best/Guaranteed Effort	No	6.1	NA
Theocharides. et al [10]	Best Effort	No	0.1	NA

5 Summary and Discussion

The work presented here is part of a long-term vision to create EMBRACE, a mixed signal hardware platform to advance large scale SNN implementations. Research approaches previously discussed, i.e. [3], [10], [11] and [12] have shown promising results in establishing the motivation to continue using the NoC paradigm as a way to overcome the interconnection problems in hardware SNNs. Nevertheless, different aspects of NoC architectures need to be explored in order to take full advantages of all its capabilities and utilisation as an interconnect fabric for SNNs platforms.

In this regards, the authors have proposed a novel adaptive NoC router architecture to alleviate the communication constraints currently experienced in the efficient realisation of SNNs in hardware. The paper demonstrates the advantages of using an adaptive NoC router architecture to improve throughput, area and power consumption. The proposed adaptive NoC router contributes to the plausibility of developing a scalable NoC-based EMBRACE SNN hardware implementation. Although having an efficient, high-throughput adaptive router is important, it is also vital that a balance between increased throughput and minimal area utilisation and power consumption is achieved. Thus, the proposed adaptive NoC router is a step forward in this direction.

Acknowledgments. Snaider Carrillo Lindado is supported by a Vice-Chancellor Research Scholarship (VCRS) from the University of Ulster. Henry Carrillo, who helped to setup some experiments using the Synopsys Design Compiler tool.

References

1. W. Gerstner, Spiking neuron models: Single neurons, populations, plasticity, Cambridge Univ Pr, 2002.
2. P. Livi and G. Indiveri, "A current-mode conductance-based silicon neuron for address-event neuromorphic systems," 2009 IEEE International Symposium on Circuits and Systems, IEEE, 2009, pp. 2898-2901.
3. 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," International Journal of Reconfigurable Computing, vol. 2009, pp. 1-13.
4. L.P. Maguire, T.M. McGinnity, B. Glackin, A. Ghani, A. Belatreche, and J. Harkin, "Challenges for large-scale implementations of spiking neural networks on FPGAs," Neurocomput., vol. 71, 2007, pp. 13-29.
5. H. Shayani, P. Bentley, and A. Tyrrell, "A Cellular Structure for Online Routing of Digital Spiking Neuron Axons and Dendrites on FPGAs," Evolvable Systems: From Biology to Hardware, Springer, 2008, pp. 273-284.
6. D. Thomas and W. Luk, "FPGA Accelerated Simulation of Biologically Plausible Spiking Neural Networks," 2009 17th IEEE Symposium on Field Programmable Custom Computing Machines, IEEE, 2009, pp. 45-52.
7. J.M. Nageswaran, N. Dutt, J.L. Krichmar, A. Nicolau, and A. Veidenbaum, "Efficient simulation of large-scale Spiking Neural Networks using CUDA graphics processors," 2009 International Joint Conference on Neural Networks, IEEE, 2009, pp. 2145-2152.
8. B. Roche, T. Mc Ginnity, L. Maguire, and L. Mc Daid, "Signalling techniques and their effect on neural network implementation sizes," Information Sciences, vol. 132, 2001, p. 67–82.
9. L. Benini and G. De Micheli, "Networks on chips: a new SoC paradigm," Computer, vol. 35, 2002, pp. 70-78.
10. T. Theocharides, G. Link, N. Vijaykrishnan, M. Irwin, and V. Srikantam, "A generic reconfigurable neural network architecture implemented as a network on chip," IEEE International SOC Conference, 2004. Proceedings., IEEE, 2004, pp. 191-194.
11. S. Philipp, J. Schemmel, and K. Meier, "A QoS network architecture to interconnect large-scale VLSI neural networks," 2009 International Joint Conference on Neural Networks, IEEE, 2009, pp. 2525-2532.
12. L.A. Plana, S.B. Furber, S. Temple, M. Khan, Y. Shi, J. Wu, and S. Yang, "A GALS Infrastructure for a Massively Parallel Multiprocessor," IEEE Design & Test of Computers, vol. 24, 2007, pp. 454-463.
13. L. McDaid, S. Hall, and P. Kelly, "A programmable facilitating synapse device," 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), IEEE, 2008, pp. 1615-1620.
14. F. Morgan, S. Cawley, B. Mc Ginley, S. Pande, L. Mc Daid, B. Glackin, J. Maher, and J. Harkin, "Exploring the evolution of NoC-based Spiking Neural Networks on FPGAs," 2009 International Conference on Field-Programmable Technology, 2009, pp. 300-303.
15. W.J. Dally and B. Towles, Principles and practices of interconnection networks, Morgan Kaufmann, 2004.
16. S. Pande, S. Carrillo, F. Morgan, S. Cawley, M. B., J. Harkin, and L. McDaid, "EMBRACE-SysC for Analysis of NoC-based Spiking Neural Network Architectures," Technical Report: Bio-Inspired Electronics and Reconfigurable Computing Research Group (BIRC), National University of Ireland, NUI Galway, Galway, Ireland, 2010.