Parallel Computing 39 (2013) 357–371

Contents lists available at SciVerse ScienceDirect

Parallel Computing

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

Fixed latency on-chip interconnect for hardware spiking neural network architectures

Sandeep Pande [a,*], Fearghal Morgan [a], Gerard Smit [b], Tom Bruintjes [b], Jochem Rutgers [b], Brian McGinley [a], Seamus Cawley [a], Jim Harkin [c], Liam McDaid [c]

[a] Bio-Inspired Electronics and Reconfigurable Computing, National University of Ireland, Galway, Ireland
[b] Computer Architecture for Embedded Systems, University of Twente, Enschede, The Netherlands
[c] Intelligent Systems Research Centre, University of Ulster, Derry, UK

ARTICLE INFO

Article history:
Received 1 August 2012
Received in revised form 16 March 2013
Accepted 8 April 2013
Available online 25 April 2013

Keywords:
Network on Chip (NoC)
Spiking Neural Networks (SNN)
Synaptic connectivity
Latency jitter

ABSTRACT

Information in a Spiking Neural Network (SNN) is encoded as the relative timing between spikes. Distortion in spike timings can impact the accuracy of SNN operation by modifying the precise firing time of neurons within the SNN. Maintaining the integrity of spike timings is crucial for reliable operation of SNN applications. A packet switched Network on Chip (NoC) infrastructure offers scalable connectivity for spike communication in hardware SNN architectures. However, shared resources in NoC architectures can result in unwanted variation in spike packet transfer latency. This packet latency jitter distorts the timing information conveyed on the synaptic connections in the SNN, resulting in unreliable application behaviour.

This paper presents a SystemC simulation based analysis of the synaptic information distortion in NoC based hardware SNNs. The paper proposes a fixed spike transfer latency ring topology interconnect for spike communication between neural tiles, using a novel time-stamped spike broadcast flow control scheme. The proposed architectural technique is evaluated using spike rates employed in previously reported mesh topology NoC based hardware SNN applications, which exhibited spike latency jitter over NoC paths. Results indicate that the proposed interconnect offers fixed spike transfer latency and eliminates the associated information distortion.

The paper presents the micro-architecture of the proposed ring router. The FPGA validated ring interconnect architecture has been synthesised using 65 nm low-power CMOS technology. Silicon area comparisons for various ring sizes are presented. Scalability of the proposed architecture has been addressed by employing a hierarchical NoC architecture.

1. Introduction

Biologically inspired artificial neural network computing techniques mimic key functions of the human brain and have the potential to offer smart and adaptive solutions for complex real world problems [1]. The organic central nervous system includes a dense and complex interconnection of neurons and synapses, where each neuron links to thousands of other neurons through synaptic connections. Computing systems based on Spiking Neural Networks (SNNs) emulate real biological

* Corresponding author.
E-mail addresses: sandeep.pande@nuigalway.ie (S. Pande), fearghal.morgan@nuigalway.ie (F. Morgan), g.j.m.smit@utwente.nl (G. Smit), t.m.bruintjes@utwente.nl (T. Bruintjes), j.h.rutgers@utwente.nl (J. Rutgers), brian.mcginley@nuigalway.ie (B. McGinley), s.cawley6@nuigalway.ie (S. Cawley), jg.harkin@ulster.ac.uk (J. Harkin), lj.mcdaid@ulster.ac.uk (L. McDaid).

0167-8191/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.parco.2013.04.010

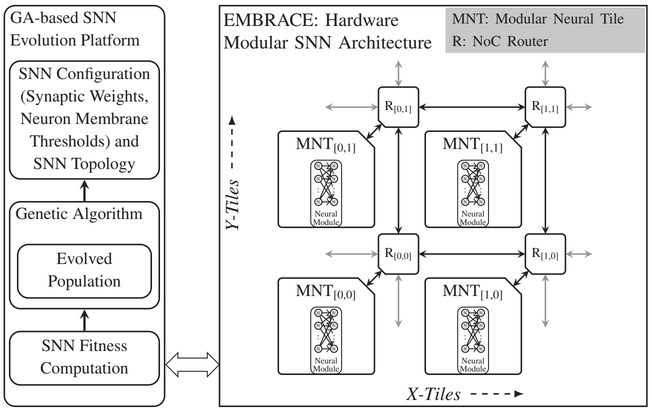


Fig. 1. EMBRACE hardware modular SNN architecture and GA-based SNN evolution platform (The Modular Neural Tile (MNT) comprises a two layered 16:16 fully connected SNN structure, packet encoder/decoder, SNN configuration and topology memory).

neural networks, conveying information through the communication of short transient pulses (*spikes*) via synaptic connections between neurons. Each neuron maintains a *membrane potential*, which is a function of incoming spikes, associated synaptic weights, current membrane potential, and the membrane potential *leakage coefficient* [2,3]. A neuron *fires* (emits a spike to all connected synapses/neurons) when its membrane potential exceeds the neuron's firing threshold value. Hardware SNN systems offer the potential for elegant, low-power and scalable methods of embedded computing, with rich non-linear dynamics, ideally suited to applications including classification, estimation, prediction, dynamic control and signal processing [4,5]. Efficient implementation of hardware SNN architectures for real-time embedded systems is primarily influenced by neuron design, scalable on-chip interconnect architecture and SNN training/learning algorithms [6].

Packet switched Network on Chip (NoC) architectures have recently been proposed as the spike communication infrastructure for hardware SNNs, where data packets containing spike information are routed over a network of routers. The NoC based synaptic connectivity approach provides flexible inter-neuron communication channels, scalable interconnect and connection reconfigurability [7,8]. The authors have investigated and proposed EMBRACE[1] (illustrated in Fig. 1)as an embedded computing element for the implementation of large scale SNNs [9–11]. A Modular Neural Tile (MNT) architecture has been proposed for reducing the silicon area, by using a combination of fixed and configurable synaptic connections [11]. The proposed MNT comprises a 16:16 neuron fully connected hardware SNN structure. MNTs are integrated in a two dimensional mesh topology packet switched NoC interconnect to form a hardware modular SNN architecture. Spike communication between the MNTs is achieved by routing spike information within spike data packets over the network of routers. The NoC architecture uses XY routing scheme and supports unicast packet flow control, where each spike packet contains destination synapse information for a single spike and is routed independently. A GA-based SNN evolution platform evolves SNN applications by finding a correct set of synaptic weights and neuron threshold potentials. A number of benchmark SNN applications such as XOR data classifier, inverted pendulum controller, Wisconsin breast cancer dataset classifier and robotic controllers have been successfully implemented on the EMBRACE-FPGA prototype [8,10,11].

Packet transfer via shared on-chip resources within a NoC based hardware SNN results in different latency values for each NoC packet. The NoC packet latency variation (jitter) alters the arrival timing of spike packets at the destination neurons, distorting the encoded information within the SNN. While a NoC based hardware SNN can be trained to produce correct application behaviour, NoC traffic dependent packet latency jitter can impact the accuracy of the SNN applications and potentially cause an application to fail. Providing guaranteed packet transfer latency is difficult in NoC based SNN architectures due to the large number of virtual synaptic connections. In past few years, research in NoC architectures have been primarily focused on providing statistical guarantees (or soft bound QoS) for large streaming data traffic [12,13]. However NoC architectures that can provide fixed packet transfer latency for a large number of virtual connections have not been explored.

This paper simulates (using clock cycle accurate SystemC models) and analyses the synaptic information distortion in NoC based hardware SNNs due to packet transfer latency jitter caused by the NoC communication infrastructure. The paper proposes a fixed spike transfer latency ring topology interconnect for spike communication between neural tiles using a novel timestamped spike broadcast flow control scheme. The proposed architectural technique is evaluated using spike rates employed in previously reported hardware SNN applications. Results indicate that the proposed interconnect offers fixed spike transfer latency and eliminates the associated information distortion. The paper presents the micro-architecture of the proposed ring router. The ring interconnect architecture has been validated on a Xilinx Virtex-6 FPGA and synthesized using 65 nm low-power CMOS technology. Silicon area comparisons for various ring sizes are presented. Limitations on the

[1] **EM**ulating **B**iologically-inspi**R**ed **A**r**C**hitectures in hardwar**E**.

scalability of the proposed ring architecture and selection of the optimal ring size based on spike rate resolution and hardware resources are discussed. Scalability of the proposed architecture has been addressed by employing a hierarchical NoC architecture.

The structure of the paper is as follows: Section 2 reviews the NoC architectures suitable for hardware SNNs. Section 3 presents simulation based analysis of the effect of latency jitter in NoC based hardware SNN architectures, and discusses the impact of the noise introduced by latency jitter on SNN applications. Section 4 presents the spike flow control of the proposed interconnect architecture and the detailed micro-architecture of the proposed ring router. Section 5 presents ring interconnect spike transfer latency results and discusses the limitations on the scalability of the proposed ring architecture. The section analyses the optimal ring size for practical spike rate encoding resolution and hardware implementation, and presents hardware resource utilisation results. Section 6 concludes the paper.

2. NoC architectures for hardware SNNs

The biological nervous system exhibits direct interconnection between neurons, where spikes stream through fixed length axons ensuing deterministic spike transfer latency and integrity of synaptic information. For biologically inspired SNN systems, the spike communication infrastructure should provide fixed spike transfer latency to ensure integrity of information. Network on Chip (NoC) architectures have been proposed as a promising solution for multi-core System on Chip (SoC) systems [14,15]. This section reviews related work on NoC based communication architectures for hardware SNN systems. This review focuses on the NoC architectures which can provide QoS for latency management and are suitable as hardware SNN interconnect.

Information in SNNs is contained within the timing of spikes. Hence the NoC spike communication infrastructure should ideally support fixed latency spike transfer in order to maintain the integrity of the encoded information. Computationally powerful hardware SNN architectures are characterised by thousands of neurons and millions of synapses [16–23]. Hence, the NoC spike communication infrastructure should support millions of virtual synaptic communication channels. Localised multicast communication schemes closely resemble the connectivity patterns typically observed in SNN application topologies and NoC architectures supporting such connectivity patterns are especially suitable for hardware SNN architectures [7]. Information distortion in SNN structures is directly proportional to spike transfer latency jitter. For practical hardware SNN implementations, the spike transfer latency can be longer than the minimum (or best case) latency, though the latency jitter should be as low as possible.

A packet switched NoC architecture can provide suitable communication infrastructure for hardware SNNs, offering scalable, parallel, distributed communication channels with flexible connection reconfigurability [24,9,7]. A NoC architecture suitable for hardware SNNs should:

– maintain the integrity of information encoded within spike timings
– support a large number of virtual synaptic communication channels
– support connection topologies that closely resemble neural connectivity patterns

In summary, the NoC architectures for hardware SNN architectures should provide a large number of virtual synaptic communication channels and localised multicast connectivity patterns, while offering low spike transfer latency jitter.

The Æthereal NoC architecture [25], developed specifically for streaming media applications, combines both Guaranteed Services (GS) and Best Effort (BE) services in a single router design. The architecture aims to offer guaranteed services such as uncorrupted, lossless, ordered data delivery, guaranteed throughput and bounded latency essential for real-time embedded multi-media applications. $\times$*pipes* [26], a scalable and high performance NoC architecture, consists of a library of synthesisable soft macros to realise latency insensitive on-chip communication. The router design is deeply pipelined and the NoC architecture is optimised for high throughput and low latency. The QNoC architecture [27] comprises a two-dimensional planar mesh topology NoC with fixed shortest path multi-class wormhole routing and provides statistical guarantees for packet communication. The QNoC architecture defines various service levels such as *Real-Time Service Level* which guarantees bandwidth and latency essential for real-time applications, such as streamed audio and video processing. SoCBUS [28], a circuit switched NoC architecture uses the initial phase of resource reservation using setup and acknowledge packets. The system can provide throughput and latency guarantees after successful channel setup.

Recent research indicates an increasing use of NoC architectures for hardware SNN systems [24,9,7]. A generic reconfigurable neural network architecture using a packet switched NoC communication infrastructure is reported in [24]. The architecture consists of a 2-D torus topology NoC, where each router serves four neurons and offers monolithic connectivity. An FPGA-based mesh topology NoC, using XY unicast routing for a clustered neural network, has been proposed in [29]. A unicast communication scheme is a limiting factor in large hardware neural network implementations because of the large number of synaptic connections. A theoretical and comparative analysis of interconnect architectures for hardware SNN systems is reported in [7]. The paper argues the use of packet switched mesh topologies over fat tree, point-to-point and shared bus topologies. Due to the connectivity patterns observed in neural topologies, multicast communication schemes outperform unicast and broadcast. A bufferless ring topology NoC architecture has been proposed, which aims to achieve low packet transfer latency while maintaining a small silicon footprint [30]. Although, the ring topologies offer deterministic hop count between nodes, the inherent connectivity pattern limits its scalability, affecting the network performance [31]. However, ring topology offers a simple and compact architecture suitable for hardware implementation. This paper presents an eight node ring topology interconnect with a fixed spike transfer latency flow control. The proposed eight neural tile ring interconnect integrates as an element within a mesh topology NoC forming a scalable, modular hardware SNN architecture.

3. Information distortion in NoC-based hardware SNN architectures

This section analyses the synaptic information distortion in SNN structures due to packet transfer latency jitter caused by the NoC communication infrastructure, using clock cycle accurate simulation models and discusses impact of the noise introduced by latency jitter on SNN applications.

3.1. Spiking neural network information coding

Understanding biological neuronal coding is one of the fundamental issues in neuroscience. Traditional hypothesis suggests that the neural information is encoded in the mean firing rate of neurons, whereas experimental results on response time of humans (for a number of senses) suggest that the neuronal information is encoded in the relative timing between the spikes [2,3,32,33]. Based on these findings, information encoding in artificial SNN systems can be broadly categorised in the following two categories:

1. **Rate Coding** technique is based on encoding the SNN information as the 'mean firing rate' of neurons. Based on different notions of the mean firing rate, this category is further subdivided in three averaging procedures, namely *Rate as a Spike Count*, *Rate as a Spike Density* and *Rate as a Population Activity* [34]. For practical SNN systems, rate coding encodes spike count within a fixed time interval (or sampling time window as depicted in Fig. 3). The time interval or sampling time of the SNN is often determined by the application response time and stimulus encoding resolution requirements. Spike rate coding technique is a popular choice for embedded applications due to its simplicity [8,10,11].
2. **Temporal Coding** technique encodes information based on the 'precise spike timings'. For practical SNN systems, this coding technique can be implemented as *Time to First Spike*, *Spike Phase Encoding* and *Spike Correlation Encoding,thorpe96*. The absence of a fixed sampling time reduces the response time of this coding technique and speeds-up the application, but the implementation complexity limits its use in practical SNN systems.

Variable spike transfer time between neurons distorts the information flowing in SNNs employing either rate coding or temporal coding. This paper presents an elaborate analysis of spike latency variation on the information flowing in SNNs employing rate coding. Due to the dependency on exact spike timings, temporal coding is highly susceptible to spike latency variations.

3.2. Latency jitter in the EMBRACE mesh topology NoC architecture

Variation in packet transfer latency resulting from the spike communication interference introduced by the reported EMBRACE NoC architecture has been modelled using EMBRACE-SysC clock cycle accurate SystemC simulation models [35]. The mean (μ_l) and standard deviation (σ_l) values for packet transfer latency (l) are used to analyse the noise introduced in SNN structures by the latency jitter. NoC paths with multiple hops (through a number of routers) are exercised with a fixed rate spike packet stream. The intermediate NoC routers are loaded with additional spike packet traffic to mimic traffic conditions in hardware SNNs. Fig. 2 illustrates the packet transfer latency variations (μ_l and σ_l) simulated for the multi-hop NoC paths under various traffic density conditions (i.e. bandwidth utilisation values of intermediate NoC routers). The mean latency plot (Fig. 2a) illustrates slow variations in the packet latency, while the latency standard deviation plot (Fig. 2b) illustrates dispersion of the spike packets received at destination neural tiles due to NoC traffic conditions. After 100% bandwidth utilisation, the intermediate NoC routers are unable to handle the traffic and therefore drop spike packets. The rate of change of mean packet latency (μ_l) increases significantly after 100% NoC traffic density. The latency standard deviation (σ_l) increases and saturates at 100% traffic density conditions in the network.

3.3. Impact of spike latency jitter on SNN applications

The packet switched NoC spike communication infrastructure cannot guarantee a precise spike transfer time for a hardware SNN system comprising a large number of virtual synaptic connections. This makes it very difficult to support temporal coding. The EMBRACE hardware SNN employs spike rate coding scheme, where information is encoded as the number of spikes in a fixed Sampling Time Window (STW = 1 ms). This technique offers practical implementation by allowing a sampling time window of a few milliseconds, required by real-life embedded applications. Benchmark SNN control and classifier applications (such as inverted pendulum controller, two-input XOR and Wisconsin breast cancer dataset classifier) have been successfully evolved on EMBRACE FPGA prototype using spike rate coding [8,10,11]. The remainder of this section analyses the impact of spike packet latency variations (due to the packet switched NoC) on information processed in SNN

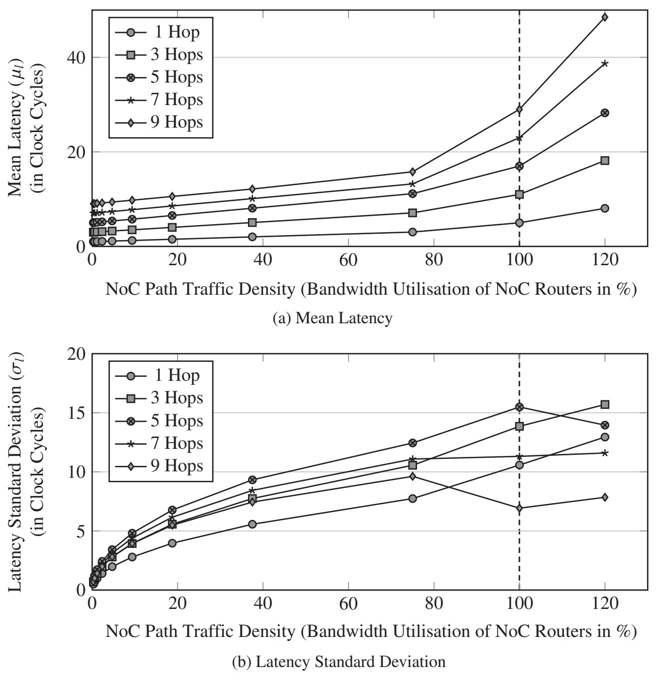


Fig. 2. Mesh topology NoC packet transfer latency variations (Mean and standard deviation of latency is measured in terms of clock cycles).

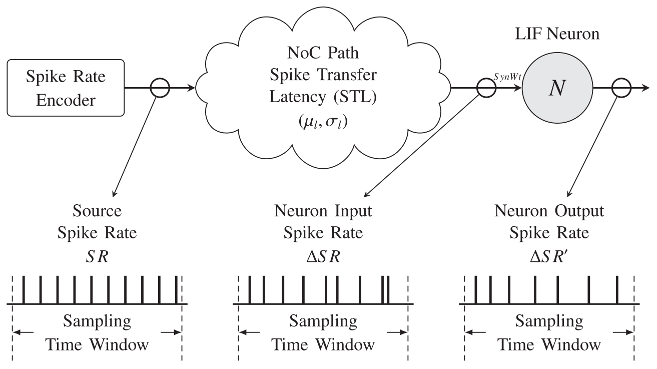

Spike Rate Encoder
NoC Path Spike Transfer Latency (STL) (μ_l, σ_l)
LIF Neuron
SynWt
N
Source Spike Rate SR
Neuron Input Spike Rate ΔSR
Neuron Output Spike Rate $\Delta SR'$
Sampling Time Window

Fig. 3. Information distortion in LIF neuron due to spike transfer latency variations

structures using simulations. The mean (μ_l) and standard deviation (σ_l) values of spike packet transfer latency (l) shown in Fig. 2 are used for the analysis.

Fig. 3 illustrates the distortion of information in a Leaky-Integrate-Fire (LIF) neuron [3] due to NoC path spike transfer latency variations. The spike rate encoder generates a fixed Spike Rate (*SR*). Spikes traverse through the packet switched NoC before reaching the neuron synaptic input. Due to the NoC path Spike packet Transfer Latency (STL) variation (μ_l, σ_l), the source spike rate *SR* deviates to ΔSR at the input of the LIF neuron. The input spikes update the neuron membrane potential by the associated synaptic weight (*SynWt*). Thus, the altered input spike rate (ΔSR) translates into the neuron membrane potential variation, resulting in additional error in the neuron output spike rate ($\Delta SR'$). Fig. 4 illustrates the LIF neuron output spike rate deviation due to input NoC path spike transfer latency jitter. An input spike event on a large synaptic weight alters the neuron membrane potential significantly and thus has a larger effect on the output spike rate changes due to jitter. The experiment highlights effect of NoC packet latency jitter on the SNN information distortion. A similar setup is used for analysis of the impact of latency jitter on SNN information in next set of experiments.

Both excitatory (positive) and inhibitory (negative) synaptic weight values affect the output spike rate of a LIF neuron due to input spike rate variations. Fig. 5 shows the comparison of LIF neuron output spike rate deviation (ΔSR) due to the input spike rate jitter and the excitatory/inhibitory synaptic weight values. Since the threshold potential of a LIF neuron can only be positive, large inhibitory synaptic weight values dampen the output spike rate significantly, resulting in a large variation

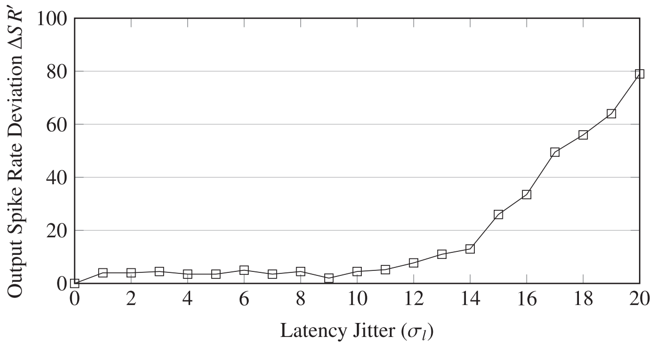


Fig. 4. LIF neuron output spike rate deviation due to latency jitter (Spike rate deviation ΔSR is measured as the difference in number of spikes within a STW of 1 ms. Constant Synaptic Weight $SynWt = 3.5$).

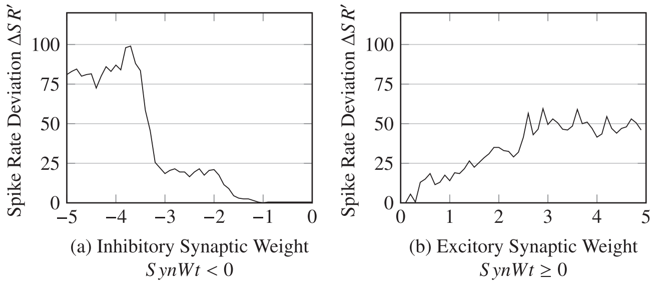


Fig. 5. Spike rate deviation (ΔSR) due to latency jitter and synaptic weight (constant latency jitter $\mu = 15$)(Spike rate deviation ΔSR is measured as the difference in number of spikes within a STW of 1 ms).

in the output spike rate due to input spike rate jitter. Similarly, large excitatory synaptic weight values cause immediate firing of the neuron resulting in significant spike rate deviation.

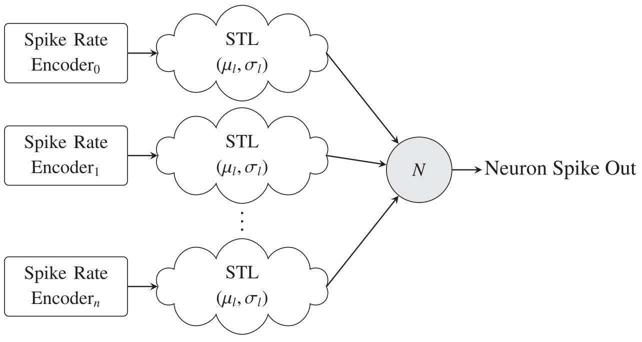
Each synaptic input experiencing spike rate jitter contributes to output spike rate deviation (proportional to the associated synaptic weight). Multiple synaptic inputs of a LIF neuron have a combined effect on the output spike rate. Fig. 6a shows the SystemC simulation setup used to measure the output spike rate error of a LIF neuron due to multiple synaptic inputs with associated spike transfer latency jitter. Fig. 6b shows the output spike rate error of a LIF neuron for a range of spike latency jitter values on multiple synaptic inputs. The output spike rate error of a LIF neuron is directly proportional to latency jitter (σ_l). Low jitter values have negligible effect on the output spike rate, since the multiple input spike events cancel the minor variations in membrane potential. As the jitter increases, the changes in membrane potential are significant and have a large impact on output spike rate.

SNN application topologies consists of multiple feed-forward and feed-back (recurrent) neuron layers. Fig. 7a shows the SystemC simulation setup used to measure the spike rate error due to cascaded layers in a feed-forward SNN topology. Fig. 7b illustrates the spike rate error in a multi-layer feed-forward SNN topology. Latency jitter has a cascading effect on the spike error rate in multi-layered SNN structures for various jitter values (μ_l, σ_l). As the information is processed in each SNN layer, the corresponding spike rate is altered due to spike transfer latency jitter. In some cases, the effect of large spike transfer latency is marginally cancelled by low spike transfer latency in the next layer. This can be seen as a slight reduction in the spike rate error in the third layer in Fig. 7.

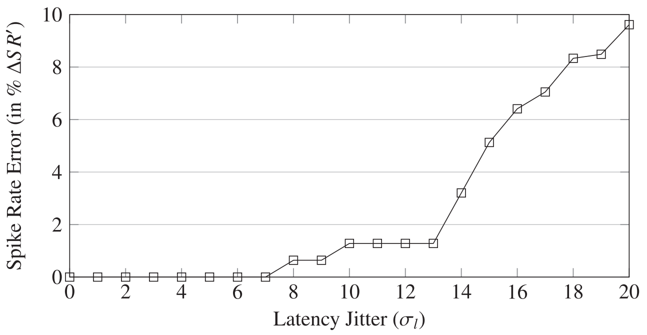
Adaptive properties of SNNs are evident from the negligible spike rate error observed at low jitter values. However, large latency jitter observed at high network traffic in the NoC distorts the SNN information considerably. This behaviour, illustrated in Figs. 4, 6 and 7, shows the traffic dependent behaviour of the SNN structure within the NoC based hardware SNN architectures and can result in erroneous application behaviour. Practical SNN systems use spike rates that are much lower than the system clock frequency. Assuming that the SNN application spike rate encoding requirement is 1024 spikes in a STW of 1 ms, the maximum spike injection rate is one spike in 196 clock cycles for the system clock frequency of 200 MHz [8,11]. Also, increasing the system clock frequency results in slower spike rates for the given encoding resolution. These properties of the SNN application spike traffic can be used to design a fixed latency spike communication infrastructure.

4. Ring topology interconnect architecture

Fixed spike transfer latency interconnect supporting a large number of virtual synaptic connections is the key for accurate and reliable operation of applications executing on packet switched NoC-based hardware SNN architectures. This section



(a) LIF Neuron Multiple Synaptic Inputs Jitter Measurement Setup



(b) Spike Error Rate for LIF Neuron Multiple Synaptic Inputs (μ_l = 30)

Fig. 6. LIF neuron multiple synaptic paths jitter (Spike rate *SR* is measured as number of spike in a STW of 1 ms).

presents the proposed ring topology interconnect architecture for hardware SNNs. Spike packet flow control of the proposed NoC architecture, supporting fixed spike transfer latency is discussed. The section also presents the detailed micro-architecture of the proposed ring router.

4.1. Fixed latency spike flow-control

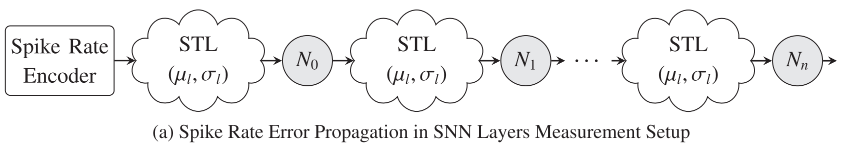
The essential elements for designing a fixed latency, packet switched spike communication interconnect are:

– connection topology offering fixed packet transfer latency between the source–destination nodes [31]
– deterministic packet transmission and reception scheduling in the source and destination routers respectively

Network topologies with appropriate flow control schemes exhibiting these properties, can offer fixed packet transfer latency. Extending this flow control to support broadcasting can efficiently support the bandwidth requirements of localised, high density synaptic connections observed in modular hardware SNNs. This section describes a ring topology with unidirectional, broadcast packet flow control to achieve fixed spike transfer latency. The unidirectional ring topology offers simple and compact router design and allows simplified placement on the silicon die. Other network topologies such as star, mesh and torus (with minimal routing) can also be used with suitable modifications in the proposed fixed latency flow control.

The flow control of the proposed ring topology interconnect treats each spike event timing separately, maintaining the clock cycle count of the spike event occurrence. The recorded spike event clock cycle count is preserved throughout the spike packet flow control. At the destination ring router, the spike event occurrence clock cycle count is used to send out the spike at the precise clock cycle (with respect to spike event occurrence) offering fixed spike transfer latency and maintaining the integrity of the SNN information flow.

Fig. 8 illustrates the proposed ring topology interconnect (in an eight node configuration) with unidirectional packet flow control for hardware modular SNN architectures. The proposed interconnect comprises routers connected in the unidirectional ring topology, where each router receives a spike packet from the previous router and sends a spike packet to the next router. The MNTs interface with ring routers for spike communication. Each router buffers the spike events generated by the attached MNT and encodes them into spike packets. Generated spike packets are processed and forwarded to the next router in a single clock cycle. Full rotation of the spike packet on the ring ensures broadcast packet flow control. The full *Rotation Cycle (RC)* of the ring (i.e. the number of clock cycles required for a generated packet to traverse all the ring routers and arrive at the source router) is equal to the number of nodes in the ring ($RC = R$).


Spike Rate Encoder
STL
(μ_l, σ_l)
N_0
N_1
N_n
(a) Spike Rate Error Propagation in SNN Layers Measurement Setup

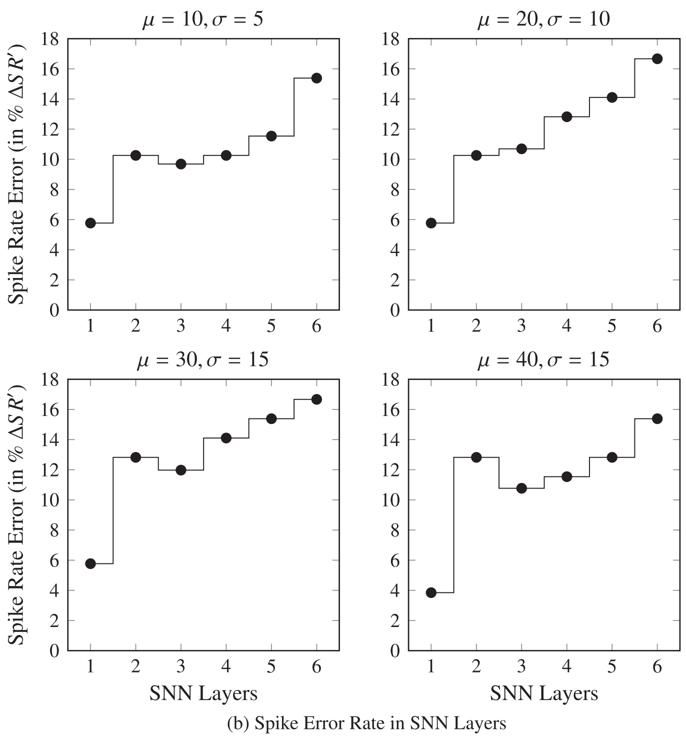


Fig. 7. Spike rate error propagation in SNN layers (Spike rate *SR* is measured as number of spike in a STW of 1 ms).

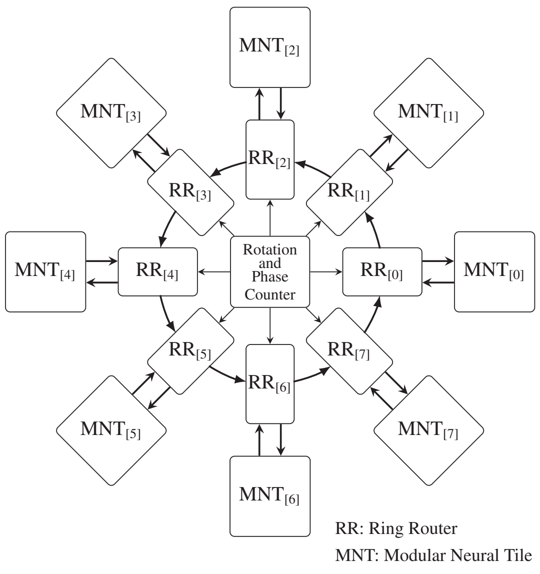


Fig. 8. Ring topology interconnect architecture (Number of ring nodes $R = 8$).

The ring interconnect operates in two phases; namely the *Insert Phase (IP)* and *Forward Phase (FP)*. Fig. 9 illustrates the sequencing of insert and forward phases of the ring, based on the clock cycle count. The ring operates in the *insert phase* for every natural number multiple of the number of ring nodes ($n = \mathbb{N}R$). For all remaining clock cycles ($n \neq \mathbb{N}R$), the ring

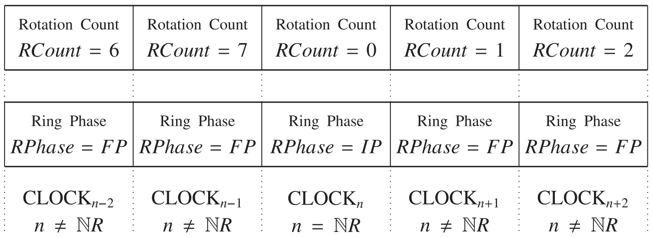


Fig. 9. Rotation count and Insert (*IP*)/Forward (*FP*) Phases of the ring (Number of ring nodes $R = 8$).

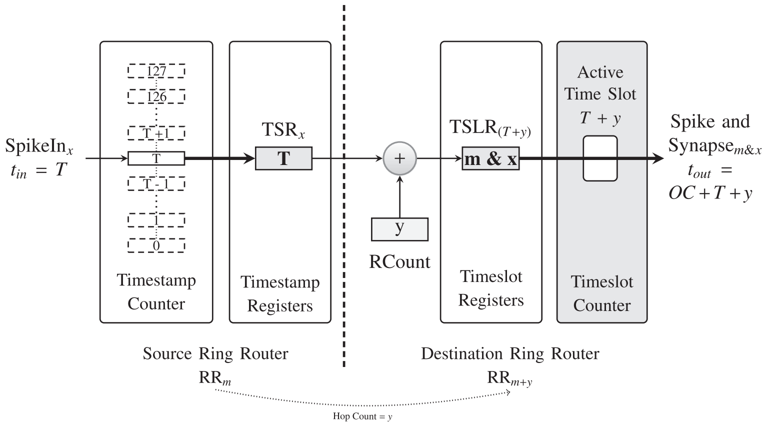


Fig. 10. Fixed latency spike packet flow control.

operates in the *forward phase*. Thus, the rotation cycle consists of one *insert phase* and $R - 1$ *forward phases*. Phase synchronisation of the ring is maintained by the global *Rotation and Phase Counter*. The phase counter maintains the clock cycle count and controls the ring operation phase. For both phases, the packet received by the router from the previous router is processed to retrieve the spike information. During an *insert phase*, the router outputs a new spike packet to the next router in the ring. During a *forward phase*, the received packet is forwarded to the next router in the ring.

Fig. 10 illustrates the spike flow control implemented on the proposed ring topology interconnect. The ring router uses a round-robin scheduling policy for transmitting spike packets corresponding to spike inputs ($\text{SpikeIn}_{[0:15]}$). A new spike packet is generated during each *insert phase*. To serve the 16 spike inputs (from the attached MNT), the ring router requires 16*RC* clock cycles (i.e. 128 clock cycles for an eight node ring configuration). This is defined as the full *Operating Cycle (OC)* of the ring. Each router maintains a timestamp counter which counts the clock cycles within an operating cycle. A valid spike event on a spike input (SpikeIn_x) is timestamped and buffered in the corresponding Timestamp Register (TSR_x). During an *insert phase*, the TSR_x value (selected by the round-robin sequence) is encoded in a new spike packet for the spike input *x*, and is forwarded to the next router in the ring.

Each ring router decodes the input spike packets and buffers the spike events for processing at precise clock cycle based on the timestamp value. On receipt, the timestamp value in the spike packet is incremented by the rotation count to account for the packet transfer latency between the corresponding source–destination router pair. The spike event is then buffered in the Time Slot Registers (TSLR) on the resulting timestamp value. A timeslot counter counting the clock cycles within the operating cycle is used to select and process spike events from the TSLRs. The TSLRs contain 128 registers for the eight node ring configuration (16*RC* registers), each corresponding to a time slot in the ring operating cycle. The spike events buffered in TSLRs are delivered during the next operating cycle. Thus the spike generated at $t_{in} = T$ in the source router, is delivered at $t_{out} = OC + T + y$. Thus the flow control offers fixed spike transfer latency of one operating cycle and the source–destination hop count ($OC + y$) for input spikes rates of $ISI \geqslant OC$ (where *ISI* is inter spike interval in number of clock cycles.

4.2. Ring router micro-architecture organisation

The ring routers facilitate spike communication between the MNTs in the ring. Fig. 11 illustrates the ring router and MNT interface. The ring router packet sender subsystem buffers the spikes generated by the attached MNT and inserts spike packets on the ring. The ring router packet receiver subsystem decodes and buffers the received spike information and sends the spikes to the attached MNT. Fig. 12 illustrates the micro-architecture organisation of the proposed ring router RTL design, comprising (a) *Packet Sender* and (b) *Packet Receiver* subsystems.

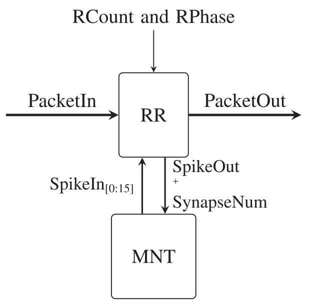


Fig. 11. Ring router and MNT interface.

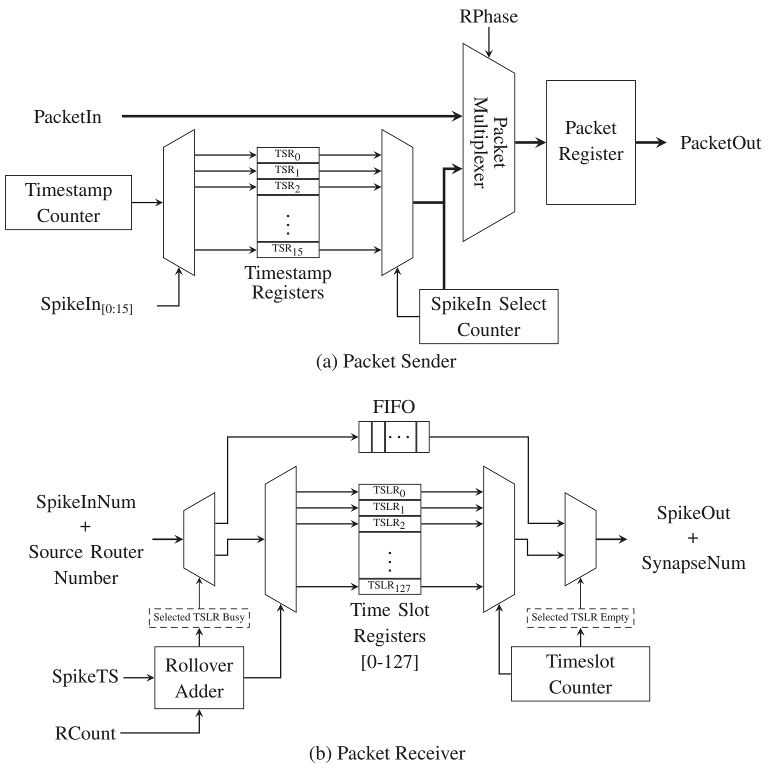


Fig. 12. Ring router micro-architecture organisation.

(a) **Packet Sender:** The phase input (RPhase), determines the operating phase of the ring router. During a forward phase, the packet multiplexer selects the input packet from the previous router. During an insert phase, a new spike packet generated by the router is selected. The selected spike packet is buffered in the packet register and passed to the next router in the ring. On receiving a valid spike on the spike input (SpikeIn$_x$), the current timestamp counter value is stored in the corresponding Timestamp Register (TSR$_x$) along with a valid bit. The packet sender uses a round-robin policy for transmitting the output spike packet for spike inputs (SpikeIn$_{[0:15]}$). The SpikeIn Select Counter is incremented during each insert phase and a new spike packet comprising the Packet Valid, Spike Timestamp and Source Spike Number is generated. Fig. 13 illustrates the packet structure for the proposed ring interconnect.

(b) **Packet Receiver:** The packet receiver subsystem decodes and buffers valid input spike packets (*PacketVal* = '1') and forwards the output spike pulse along with synapse number to the MNT. The Spike Timestamp (*SpikeTS*), the Rotation Count (*RCount*) and the router number (*RR$_n$*) present in each router are used to calculate the resulting timeslot for the output spike pulse. The *rollover adder* increments the timestamp value by the rotation count value. If the result is larger than the maximum number of timeslots in the operating cycle, the sum is rolled over to the next operating cycle. The rollover adder output selects the Timeslot Register (TSLR) for storing the SpikeIn Number (*SpikeInNum*) from the input spike packet, source router number (calculated from destination router number and RCount), and asserts the

B_{11}	B_{10}-B_4	B_3-B_0
PacketVal Packet Valid (1-bit)	***SpikeTS*** Spike Timestamp (7-bits)	***SpikeInNum*** Spike Input Number (4-bits)

Fig. 13. Ring interconnect packet structure.

Busy bit. If the selected TSLR is already occupied (*Busy* = ’1’), the *SpikeInNum* and source router number values are pushed into the FIFO.The Timeslot Counter continuously counts clock cycles within the operating cycle and keeps track of the current time slot. The timeslot count is used to select the TSLR and retrieve the *SpikeInNum* and source router number. If the selected TSLR is empty (*Busy* = ‘0’), the next available entry from the FIFO is retrieved. The *SynapseNum* is computed by concatenating the source router number and *SpikeInNum*, and is sent to the attached MNT along with a generated spike pulse (*SpikeOut*).

Multiple spike events that are destined for the same TSLR are buffered in the FIFO for delivery during vacant time slots, which are indicated by an empty TSLR. This untimed delivery of spikes results in variable spike transfer latency. As the spike rates for practical SNN applications is considerably lower than the worst case spiking rates, the probability of the TSLR collisions is low.

5. Results and discussion

The proposed ring topology interconnect has been modelled as a clock-cycle accurate model in SystemC. Performance of the ring architecture has been measured using EMBRACE-SysC design exploration framework for hardware SNN architectures [35]. This section presents spike transfer latency and hardware implementation results for the proposed ring interconnect. Scalability of the proposed ring interconnect has been addressed by employing a hierarchical NoC architecture.

5.1. Ring topology interconnect spike transfer latency performance

Performance of the proposed ring topology interconnect has been evaluated using worst case spike traffic observed in SNN applications [8,11]. The measurement setup uses spike rate encoders (instead of MNTs as shown in Fig. 8) feeding constant spike streams to each router in the ring. Each spike rate encoder has 16 spike outputs connected to the spike inputs ($\text{SpikeIn}_{[0:15]}$) of the corresponding ring router. The frequency of spikes on each of the spike inputs (SpikeIn_x) is $\frac{1}{ISI}$; where *ISI* is *Inter Spike Interval* in clock cycles. Each ring router can receive a spike every $ISI_{min}/16$ clock cycles from the attached MNT, resulting in a maximum spike rate of $16F/ISI_{min} = 25 \times 10^6$ spikes per second (where system clock frequency F = 200 MHz and ISI_{min} = 128 clock cycles). Overall the ring generates, transfers and consumes $(16F\mathbb{N}R)/ISI_{min} = 200 \times 10^6$ spikes per second. The proposed ring interconnect supports high spike density communication, with each ring router able to deliver a spike on every clock cycle. The previously reported EMBRACE mesh topology NoC employing unicast flow control scheme cannot offer such performance.

Table 1 illustrates the mean spike transfer latency (μ_l) for multi-hop spike traffic on the proposed ring topology interconnect in an eight node configuration. The interconnect offers fixed latency for $ISI \geqslant OC$. As the spike rate reaches its upper limit ($ISI < OC$), the generated spike events are overwritten on TSRs (in the packet sender subsystem within the source ring router), resulting in loss of spikes. Fig. 14 illustrates the increase in spike loss due to the increase in input spike rate. Also, as the spike rate exceeds the upper limit ($ISI < OC$), the probability of received spike events destined for the same time slot increases, resulting in collisions on the destination TSLRs. The packet receiver subsystem within the destination ring router buffers these spike events (causing collisions on the TSLR) in a FIFO. Due to the unavailability of time slots and untimed delivery of spike events from the FIFO, the spike transfer latency variations increase considerably. Fig. 15 illustrates the spike

Table 1
Mean spike transfer latency (μ_l) for multiple hop spike traffic on the proposed ring topology interconnect.

Inter spike interval	1-Hop	2-Hop	3-Hop	4-Hop	5-Hop	6-Hop	7-Hop	Rotation full ring 8-Hop
2048	129	130	131	132	133	134	135	128
1024	129	130	131	132	133	134	135	128
512	129	130	131	132	133	134	135	128
256	129	130	131	132	133	134	135	128
128	129	130	131	132	133	134	135	128
96	204.87	205.87	206.87	207.87	208.87	209.87	210.87	203.87
64	269.80	270.80	271.80	272.80	273.80	274.80	275.80	268.80
32	327.45	328.45	329.45	330.45	331.45	332.45	333.45	326.45

Inter Spike Interval (ISI) is specified in clock cycles between consecutivespikes and the spike transfer latency is measured in clock cycles.

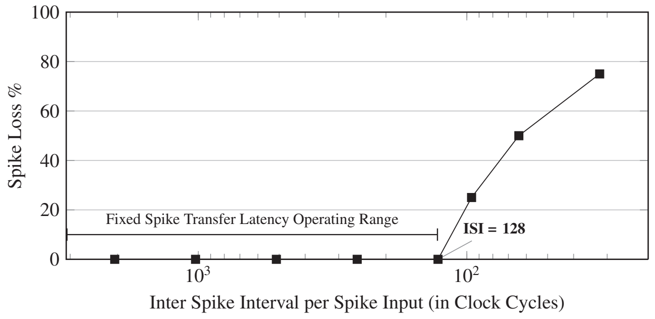


Fig. 14. Ring topology interconnect spike loss rate.

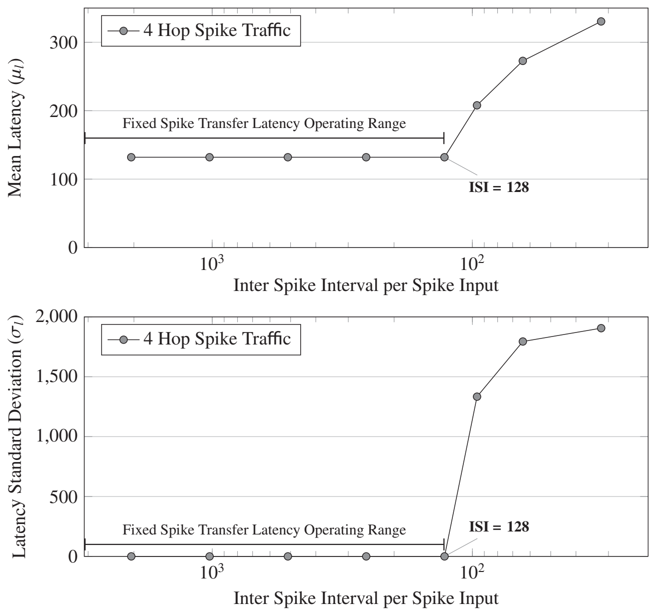


Fig. 15. Ring topology interconnect spike transfer latency variations (Inter Spike Interval is specified in clock cycles between consecutive spikes. Mean and standard deviation of latency is measured in terms of clock cycles).

transfer latency variations (μ_l and σ_l) measured for the 4-hop spike traffic on the proposed ring topology interconnect. For $ISI < 128$ (i.e. the operating cycle of the eight node ring), the spike transfer latency mean and standard deviation (μ_l and σ_l) increase significantly.

The operating cycle ($OC = RSpikeIn_n$) of the ring imposes a limit on the maximum spike rate supported (for spike inputs), for the fixed spike transfer latency operation of the proposed ring interconnect. The maximum number of spikes for a sampling time window[2] of 1 ms for various ring sizes is depicted in Table 2 (Highlighted values represent the selected ring size of eight nodes). A large number of spikes within a STW allows higher resolution for spike rate encoding in a SNN application setup (Fig. 3).

5.2. Hardware implementation

The proposed ring interconnect has been implemented as a configurable design in VHDL. A number of ring configurations have been synthesised to Xilinx Virtex 6 FPGA (to validate the RTL design) and 65 nm low-power CMOS technology from STMicroelectronics for silicon area results. Table 3 shows the hardware synthesis results for the proposed ring topology interconnect where the eight ring node configuration selected for hardware implementation is highlighted. ASIC synthesis has been performed using Synopsys Design Compiler 2009-sp5 and FPGA synthesis using Xilinx XST 14.4.

The ring router architecture is primarily organised around timestamp and time slot registers, which are integral parts of the packet sender and receiver subsystems. The number of timestamp registers is determined by the number of spike inputs,

[2] Choice of sampling window time is primarily influenced by the response time requirements of the application.

Table 2
Neuron density, spike injection rate and maximum number of spikes in a Sampling Time Window (STW) of 1 ms, for various ring sizes.

Ring size	Neurons	Synapses	Minimum ISI (in clock cycles)	Maximum spikes (in a STW)
4	128	5 K	64	3125
8	**256**	**18 K**	**128**	**1562**
16	512	68 K	256	781
32	1024	264 K	512	390
64	2048	1040 K	1024	195
128	4096	4128 K	2048	97
256	8192	16448 K	4096	48

Table 3
Ring topology interconnect synthesis results (Clock frequency: 200 MHz).

Ring size	ASIC synthesis (65 nm low-power CMOS)		FPGA synthesis (Xilinx Virtex-6 FPGA)		
	Ring NoC area (in μm^2)	Ring router area (in μm^2)	Ring NoC slices	Ring router slices	Slice registers
4	40,580	10,136	1060	265	581
8	**157,700**	**19,707**	**5202**	**650**	**1175**
16	642,555	40,145	24,010	1212	2486
32	2,723,136	85,099	NA	2970	5322

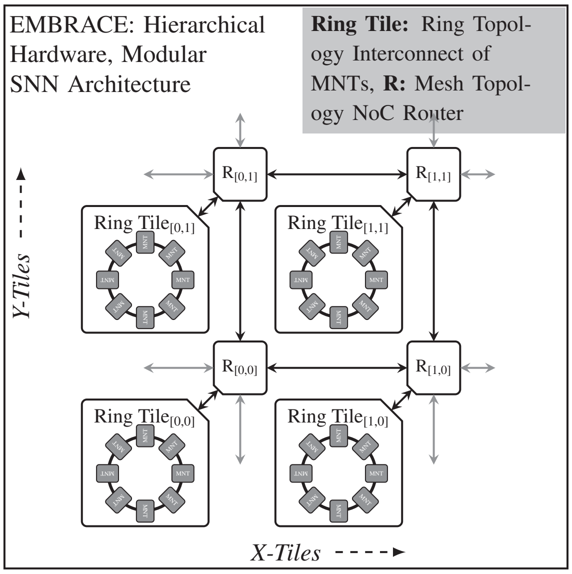

EMBRACE: Hierarchical Hardware, Modular SNN Architecture
Ring Tile: Ring Topology Interconnect of MNTs, R: Mesh Topology NoC Router
R[0,1]
R[1,1]
R[0,0]
R[1,0]
Ring Tile[0,1]
Ring Tile[1,1]
Ring Tile[0,0]
Ring Tile[1,0]
MNT
Y-Tiles
X-Tiles

Fig. 16. Hierarchical modular SNN architecture comprising ring topology interconnects within mesh topology NoC.

and the register width is defined by the operating cycle. The number of time slot registers is determined by the operating cycle, and the register width is defined by the operating cycle and the number of synapses in the ring. As the ring dimension scales, the number of synapses increases quadratically and similar change is observed in the silicon footprint of the ring router and the complete interconnect. In FPGA synthesis, the TSR and TSLR registers are mapped to slice registers resulting in a proportional increase in slice register count and associated control logic mapped onto the slices.

The previously reported EMBRACE mesh topology NoC uses unicast packet flow control for the synaptic connections between MNTs. The architecture employed SNN topology memory for storing the spike packet information for the large number of synaptic connections. The topology memory within each MNT occupies 50% of the silicon area [11]. The proposed ring interconnect employs spike broadcast flow control, where the weight values for unwanted synaptic connections are set to zero in the neuron configuration. This results in a significant reduction in the silicon footprint of the architecture.

The rotation and phase counter is replicated in each ring router to facilitate scalable hardware implementation of the interconnect. For simplicity, the rotation and phase counter is shown as a central unit in Fig. 8.

Table 4
Hierarchical modular SNN architecture FPGA synthesis results (Xilinx Virtex-6)

RTL Entity	Slices	Slice registers	DSP48E1	BRAM
Neuron	8	16	1	0
Modular Neural Tile	313	1705	32	0
Interface Tile	490	656	0	16
Ring Router	617	1175	0	1
Ring Tile	7693	22004	224	24
Mesh NoC Router	133	35	0	0

5.3. Scalability for large SNN arrays

Practical SNN systems are characterised by a large numbers of neurons and high interconnectivity through inter-neuron synaptic connections. Each of the SNN execution architectures presented in [16–23] aims for thousands of neurons and millions of synapses, which is essential for a powerful neural computing platform. Hence, scalability is an important aspect of interconnect design for hardware SNN architectures.

The spike flow control of the proposed ring topology interconnect relies on the ring size and number of spike inputs. Scaling the ring interconnect results in a quadratic increase in the synaptic density, but also increases the spike transfer latency. Scaling also adversely affects the spike rate encoding resolution (Table 2). Based on the supported synaptic density, spike rate resolution (which is suitable for practical SNN applications) and silicon area requirements, an optimal sized eight node ring interconnect with MNTs has been designed as the *Ring Tile* (Table 2).

Research shows that real world biological networks exhibit high communication locality [36,37]. The modular organisation in the human brain has been the motivation for the MNN design strategy, which suggests partitioning of application tasks into a number of subtasks [38–40]. Based on these factors, the proposed interconnect architecture can be scaled by replicating individual ring tiles connected in a mesh topology NoC architecture. The individual ring tile is made up of the ring topology interconnect of seven MNTs and one interface tile. The mesh topology NoC router bandwidth utilisation drops considerably as most of the localised, high density spike traffic is contained within the ring. This results in low spike packet latency jitter for inter ring tile spike communication.

Fig. 16 illustrates the architecture of the proposed hierarchical modular hardware SNN. The proposed hierarchical, modular hardware SNN in a two ring tile configuration comprises 448 neurons, 32 K synapses and has been synthesised on Xilinx Virtex-6 XC6VLX240T FPGA device and the FPGA synthesis results are presented in Table 4.

6. Conclusions

Spike transfer latency jitter in previously reported EMBRACE mesh topology NoC architecture for hardware SNNs has been quantified and presented as mean and standard deviation of latency over synaptic paths. The mean latency changes represent slow variations in latency, whereas the standard deviation represents latency jitter. The latency jitter in XY routing based mesh topology NoC increases with the network traffic and can affect the SNN application behaviour.

This paper analysed the effects of NoC latency jitter on the information in SNN structures. The LIF neuron output spike rate is dependent on the input spike rate, synaptic weight and associated latency jitter on the input synaptic path. The analysis shows that the LIF neuron output spike rate error is directly proportional to input spike latency jitter. Large inhibitory synaptic weight values have a bigger impact on the LIF neuron output spike rate, due to the associated jitter as compared to excitatory synaptic weight values. The synaptic weight integration within an LIF neuron cancels the minor variations (low jitter) on the multiple synaptic inputs, but the high jitter results in a steep increase in the neuron output spike rate error. The SNN information error increases due to the latency jitter in NoC, as the information is processed in cascaded SNN layers connected via NoC.

A ring topology interconnect offering a fixed spike transfer latency flow control has been presented. The proposed ring interconnect supports high spike density communication. The spike transfer latency is proportional to the ring size and number of spike inputs to each ring router. The proposed eight node ring configuration offers a fixed latency of 128–135 clock cycles between the ring nodes for $ISI \geqslant 128$ clock cycles. The fixed latency offered by the proposed ring interconnect makes the architecture suitable for employing temporal coding in SNN applications. Scaling the ring interconnect increases the number of synaptic connections quadratically, but also results in proportional increase in spike event buffer size. The ring sizes with more than eight nodes have been found to be unsuitable for applications with response time requirement of few milliseconds, due to spike rate coding constraints and limited resolution. A hierarchical, modular hardware SNN architecture comprising ring tiles integrated in a mesh topology NoC has been presented as a scalable SNN computing platform.

Future work includes validation of the proposed hierarchical hardware modular SNN architecture using real-life modular SNN applications. A modular robotic navigational controller application comprising multiple application subtasks running on MNTs within the ring topology interconnect is in development. The suitability of the proposed architecture as a hardware platform for real-life cognitive embedded applications will be analysed by measuring the application accuracy and reliability.

Acknowledgement

This research is supported by the International Centre for Graduate Education in Micro and Nano-Engineering (ICGEE), Irish Research Council for Science, Engineering and Technology (IRCSET) and Xilinx University Programme.

References

[1] S. Haykin, Neural Networks: A Comprehensive Foundation, vol. 13, Prentice Hall, 1999.

[2] W. Maass, Networks of spiking neurons: the third generation of neural network models, Neural Networks, 0893-6080 10 (9) (1997) 1659–1671.

[3] W. Gerstner, W.M. Kistler, Spiking Neuron Models: Single Neurons, Populations, Plasticity, Cambridge University Press, 2002, ISBN 9780521890793.

[4] S.M. Bohte, J.N. Kok, Applications of spiking neural networks, Information Processing Letters 95 (6) (2005) 519–520.

[5] M. Pearson, A. Pipe, B. Mitchinson, K. Gurney, C. Melhuish, I. Gilhespy, M. Nibouche, Implementing spiking neural networks for real-time signal-processing and control applications: a model-validated FPGA approach, IEEE Transactions on Neural Networks 18 (5) (2007) 1472–1487.

[6] L. Maguire, T. McGinnity, B. Glackin, A. Ghani, A. Belatreche, J. Harkin, Challenges for large-scale implementations of spiking neural networks on FPGAs, Neurocomputing, 0925-2312 71 (1–3) (2007) 13–29.

[7] D. Vainbrand, R. Ginosar, Scalable network-on-chip architecture for configurable neural networks, Microprocessors and Microsystems, 0141-9331 35 (2) (2011) 152–166.

[8] F. Morgan, S. Cawley, B. McGinley, S. Pande, L. McDaid, B. Glackin, J. Maher, J. Harkin, Exploring the evolution of NoC-based Spiking Neural Networks on FPGAs, in: International Conference on Field-Programmable Technology, 2009. FPT, 2009, pp. 300–303.

[9] J. Harkin, F. Morgan, L. McDaid, S. Hall, B. McGinley, S. Cawley, A reconfigurable and biologically inspired paradigm for computation using network-on-chip and spiking neural networks, International Journal of Reconfigurable Computing, 1687-7195 (2009) 2:1–2:13.

[10] S. Cawley, F. Morgan, B. McGinley, S. Pande, L. McDaid, S. Carrillo, J. Harkin, Hardware spiking neural network prototyping and application, Genetic Programming and Evolvable Machines, 1389-2576 12 (2011) 257–280.

[11] S. Pande, F. Morgan, S. Cawley, T. Bruintjes, G. Smit, B. McGinley, S. Carrillo, J. Harkin, L. McDaid, Modular neural tile architecture for compact embedded hardware spiking neural network, Neural Processing Letters, 1370-4621 (2013) 1–23.

[12] T. Bjerregaard, S. Mahadevan, A survey of research and practices of network-on-chip, ACM Computing Surveys 38 (1) (2006), ISSN 0360-0300.

[13] E. Salminen, A. Kulmala, T. Hamalainen, On network-on-chip comparison, in: 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools, 2007. DSD, 2007, pp. 503–510.

[14] L. Benini, G. De Micheli, Powering networks on chips, in: Proceedings of the 14th International Symposium on System Synthesis, 2001, pp. 33–38.

[15] L. Benini, G. De Micheli, Networks on chips: a new SoC paradigm, Computer, 0018-9162 35 (1) (2002) 70–78.

[16] M. Ehrlich, C. Mayr, H. Eisenreich, S. Henker, A. Srowig, A. Grubl, J. Schemmel, R. Schuffny, Wafer-scale VLSI implementations of pulse coupled neural networks, in: Proceedings of the International Conference on Sensors, Circuits and Instrumentation Systems, 2007.

[17] J. Schemmel, J. Fieres, K. Meier, Wafer-scale integration of analog neural networks, in: IEEE International Joint Conference on Neural Networks, 2008. IJCNN. (IEEE World Congress on Computational Intelligence), ISSN 1098-7576, 2008, pp. 431–438.

[18] S. Furber, A. Brown, Biologically-inspired massively-parallel architectures – computing beyond a million processors, in: Ninth International Conference on Application of Concurrency to System Design, 2009. ACSD'09, ISSN 1550-4808, 2009, pp. 3–12.

[19] E. Ros, E. Ortigosa, R. Agis, R. Carrillo, M. Arnold, Real-time computing platform for spiking neurons (RT-spike), IEEE Transactions on Neural Networks, 1045-9227 17 (4) (2006) 1050–1063.

[20] A. Upegui, C.A. Pea-Reyes, E. Sanchez, An FPGA platform for on-line topology exploration of spiking neural networks, Microprocessors and Microsystems, 0141-9331 29 (5) (2005) 211–223.

[21] M. Pearson, A. Pipe, B. Mitchinson, K. Gurney, C. Melhuish, I. Gilhespy, M. Nibouche, Implementing spiking neural networks for real-time signal-processing and control applications: a model-validated fpga approach, IEEE Transactions on Neural Networks, 1045-9227 18 (5) (2007) 1472–1487.

[22] B. Glackin, T. McGinnity, L. Maguire, Q. Wu, A. Belatreche, A novel approach for the implementation of large scale spiking neural networks on fpga hardware, in: J. Cabestany, A. Prieto, F. Sandoval (Eds.), Computational Intelligence and Bioinspired Systems, Lecture Notes in Computer Science, vol. 3512, Springer, Berlin/ Heidelberg, 2005, ISBN 978-3-540-26208-4, pp. 1–24.

[23] R. Vogelstein, U. Mallik, J. Vogelstein, G. Cauwenberghs, Dynamically reconfigurable silicon array of spiking neurons with conductance-based synapses, IEEE Transactions on Neural Networks, 1045-9227 18 (1) (2007) 253–265.

[24] T. Theocharides, G. Link, N. Vijaykrishnan, M. Invin, V. Srikantam, A generic reconfigurable neural network architecture as a network on chip, in: SOC Conference, 2004. Proceedings. IEEE International, 2004, pp. 191–194.

[25] K. Goossens, J. Dielissen, A. Radulescu, Æthereal network on chip: concepts, architectures, and implementations, IEEE Design Test of Computers, 0740-7475 22 (5) (2005) 414–421.

[26] M. Dall'Osso, G. Biccari, L. Giovannini, D. Bertozzi, L. Benini, Xpipes: a latency insensitive parameterized network-on-chip architecture for multiprocessor SoCs, in: Proceedings of the 21st International Conference on Computer Design, 2003, ISSN 1063–6404, 2003, pp. 536–539.

[27] E. Bolotin, I. Cidon, R. Ginosar, A. Kolodny, QNoC: QoS architecture and design process for network on chip, Journal of Systems Architecture, 1383-7621 50 (2–3) (2004) 105–128.

[28] D. Wiklund, D. Liu, SoCBUS: switched network on chip for hard real time embedded systems, in: Parallel and Distributed Processing Symposium, 2003. Proceedings. International, ISSN 1530–2075, 2003, pp. 8

[29] R. Emery, A. Yakovlev, G. Chester, Connection-centric network for spiking neural networks, in: Proceedings of the 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip, NOCS'09, IEEE Computer Society, 2009, ISBN 978-1-4244-4142-6, pp. 144–152.

[30] J. Kim, H. Kim, Router microarchitecture and scalability of ring topology in on-chip networks, in: Proceedings of the 2nd International Workshop on Network on Chip Architectures, NoCArc'09, ACM, 2009, ISBN 978-1-60558-774-5, pp. 5–10.

[31] W. Dally, B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003, ISBN 0122007514.

[32] W. Gerstner, A.K. Kreiter, H. Markram, A.V.M. Herz, Neural codes: firing rates and beyond, Proceedings of the National Academy of Sciences 94 (24) (1997) 12740–12741.

[33] S. Thorpe, D. Fize, C. Marlot, Speed of processing in the human visual system, Nature 381 (1996) 520–522.

[34] F. Rieke, D. Warland, R. Deruytervansteveninck, W. Bialek, Spikes: Exploring the Neural Code (Computational Neuroscience), The MIT Press, 1999, ISBN 0262681080.

[35] S. Pande, F. Morgan, S. Cawley, B. McGinley, S. Carrillo, J. Harkin, L. McDaid, EMBRACE-SysC for analysis of NoC-based Spiking Neural Network architectures, in: 2010 International Symposium on System on Chip (SoC), 2010, pp. 139–145.

[36] D.J. Watts, S.H. Strogatz, Collective dynamics of 'small-world' networks, Nature, 0028-0836 393 (6684) (1998) 440–442.

[37] S.H. Strogatz, Exploring complex networks, Nature, 0028-0836 410 (6825) (2001) 268–276.

[38] B.L. Happel, J.M. Murre, Design and evolution of modular neural network architectures, Neural Networks, 0893-6080 7 (6–7) (1994) 985–1004.

[39] G. Auda, M.S. Kamel, Modular neural networks a survey, International Journal of Neural Systems 9 (2) (1999) 129–151.

[40] Ronco, P. Gawthrop, Modular neural networks: a state of the art, Rapport Technique CSC95026, vol. 1, Center of System and Control, University of Glasgow, 1995, pp. 1–22.