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], 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, Northern Ireland, UK.*

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 alters spike timings and distorts the information conveyed on the synaptic connections in the SNN, resulting in unreliable application behaviour. This paper presents a simulation-based analysis of this synaptic information distortion in NoC based hardware SNNs.

The paper proposes a ring topology interconnect for spike communication between neural tiles, and a timestamped spike broadcast flow control scheme that offers fixed spike transfer latency. 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 synthesised using 65nm low-power CMOS technology. Silicon area comparisons for various ring sizes are presented. Limitations on the scalability of the proposed ring architecture and selection of the optimal ring size based on spike rate resolution and hardware resources are discussed. A hierarchical, mesh topology NoC architecture with a 4×8 modular neural elements supporting 896 neurons and 72K synapses on a Xilinx Virtex-6 XC6VLX240T FPGA device is presented.

Keywords: Network on Chip (NoC), Spiking Neural Networks (SNN), Synaptic Connectivity, Latency Jitter

[*] Corresponding author
Email addresses: `sandeep.pande@nuigalway.ie` (Sandeep Pande), `fearghal.morgan@nuigalway.ie`

1. Introduction

Biologically inspired artificial neural network computing techniques mimic the 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 connects to thousands of other neurons through synaptic connections. Computing systems based on Spiking Neural Networks (SNNs) emulate real biological 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. Research on biological neurons suggests that the information is encoded in the mean firing rate of the neurons, or the relative timing between the spikes [2] [3]. For SNN practical implementations, *rate based coding*, where the system inputs/outputs are encoded as distinct spike rates and *temporal based coding*, employing precise timing between the spikes are used [4] [5] [6].

Hardware SNN based 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. The 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 [7]. 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 [8] [9]. The authors have investigated and proposed EMBRACE[1] as an embedded computing element for implementation of large scale SNNs [10] [11] [12]. The proposed EMBRACE mixed-signal architecture incorporates neuron circuits interconnected using a packet switched NoC architecture.

Packet transfer on the 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, a change in NoC traffic pattern can impact the accuracy of the SNN application. The traffic dependent NoC packet latency jitter can cause an application to fail due to changes in the NoC traffic pattern. Providing guarantees for packet transfer latency is difficult in NoC based SNN architectures due to the large number of virtual synaptic connections. 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 ring topology interconnect for spike communication between neural tiles, and a timestamped spike broadcast flow control scheme that offers fixed spike transfer la-

(Fearghal Morgan), `g.j.m.smit@utwente.nl` (Gerard Smit), `t.m.bruintjes@utwente.nl` (Tom Bruintjes), `j.h.rutgers@utwente.nl` (Jochem Rutgers), `s.cawley6@nuigalway.ie` (Seamus Cawley), `jg.harkin@ulster.ac.uk` (Jim Harkin), `lj.mcdaid@ulster.ac.uk` (Liam McDaid)

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

tency. 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 synthesised using 65nm low-power CMOS technology. Silicon area comparisons for various ring sizes are presented. Limitations on the scalability of the proposed ring architecture and selection of the optimal ring size based on spike rate resolution and hardware resources are discussed. A hierarchical, mesh topology NoC architecture with a 4×8 modular neural elements supporting 896 neurons and 72K synapses on a Xilinx Virtex-6 XC6VLX240T FPGA device is presented.

The structure of the paper is as follows: Section 2 reviews the NoC architectures suited for hardware SNN implementations. Section 3 presents simulation results 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 scalability of the proposed ring architecture. The section analyses the optimal ring size for practical spike rate encoding resolution and hardware implementation. Hardware resource utilisation results are also presented. Section 6 concludes the paper and presents future work.

2. NoC Architectures for Hardware SNNs

This section reviews related work on NoC based communication architectures for hardware SNN systems. The review focuses on NoC architectures which can be used as hardware SNN interconnect and provide latency management.

Network on Chip (NoC) architectures have been proposed as a promising solution for multi-core System on Chip (SoC) systems [13] [14]. A typical NoC architecture comprises interconnected routers, which support data transfer between on-chip entities. NoC architectures can be broadly categorised as *Circuit Switched* or *Packet Switched*. Circuit switched NoC architectures provide data connectivity to SoC entities by allocating dedicated physical communication channels between the on-chip routers. Packet switched NoC architectures route individual data packets over a network of routers. A packet switched NoC architecture can provide a suitable communication infrastructure for hardware SNNs, offering scalable, parallel, distributed communication channels with flexible reconfigurability [15] [10] [8].

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

Information in SNNs is contained within spike timings. 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] [17] [18] [19] [20] [21] [22] [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 [8]. 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.

The Æthereal NoC architecture [24], 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. ×*pipes* [25], a scalable and high performance NoC architecture, consists of a library of synthesizable 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 [26] 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 [27], 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 [15] [10] [8]. A generic reconfigurable neural network architecture using a packet switched NoC communication infrastructure is reported in [15]. 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 [28]. A unicast communication scheme is a limiting factor in large hardware neural network implementations because of large number of synaptic connections. A theoretical and comparative analysis of interconnect architectures for hardware SNN systems is reported in [8]. 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 architectures have been proposed, which aim to achieve low packet transfer latency while maintaining a small silicon footprint [29]. Although, the ring topologies offer deterministic hop counts between nodes, the inherent connectivity pattern limits its scalability, affecting the network performance [30]. However, ring topology offers simple and compact architecture suitable for hardware implementation. This paper presents an eight node ring topology interconnect along 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.

The authors have investigated and proposed EMBRACE (figure 1), a NoC based hardware SNN architecture for implementation of large scale SNNs, specifically targeted at embedded applications [10] [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 [12]. The proposed MNT comprises a 16:16 neuron fully connected SNN. MNTs are integrated in a two dimensional mesh topology, packet switched NoC interconnect to form a hardware modular SNN architecture (figure. 1). Synaptic connectivity is achieved by routing unicast spike packets. A hierarchical NoC architecture has also been explored to address the scalability issues of neural interconnect architectures [31].

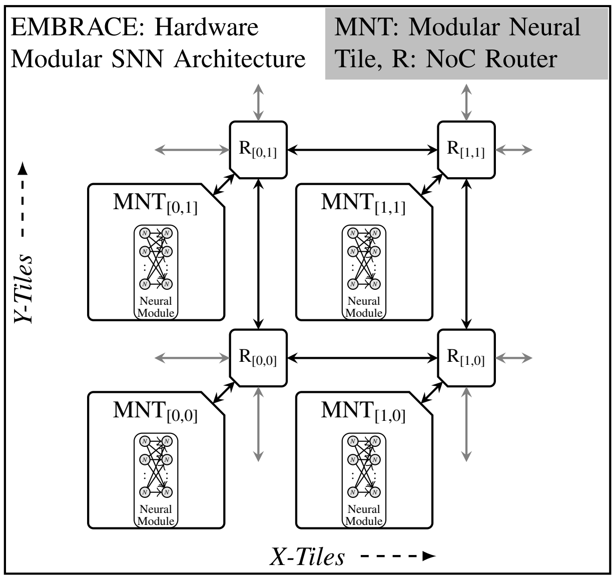


Figure 1: EMBRACE Hardware Modular SNN Architecture (The Modular Neural Tile (MNT) comprises a two layered 16:16 fully connected SNN structure, packet encoder/decoder, SNN configuration and topology memory)

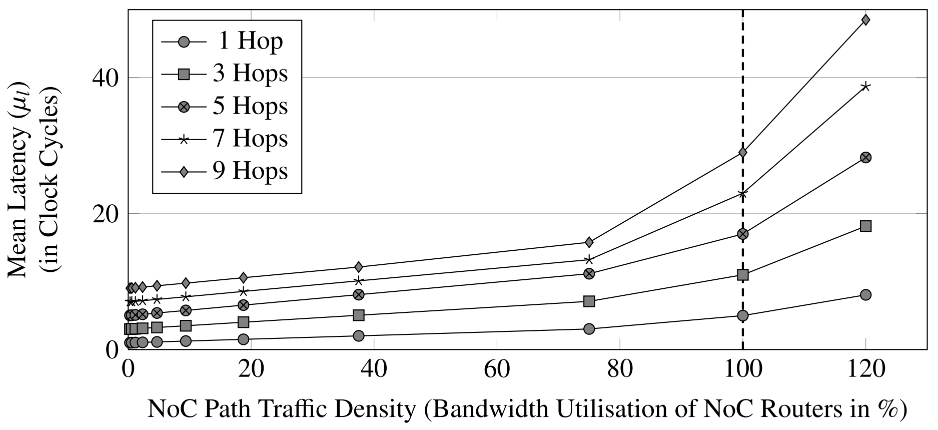
3. Information Distortion in NoC-based Hardware SNN Architectures

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 the impact of the noise introduced by latency jitter on SNN applications.

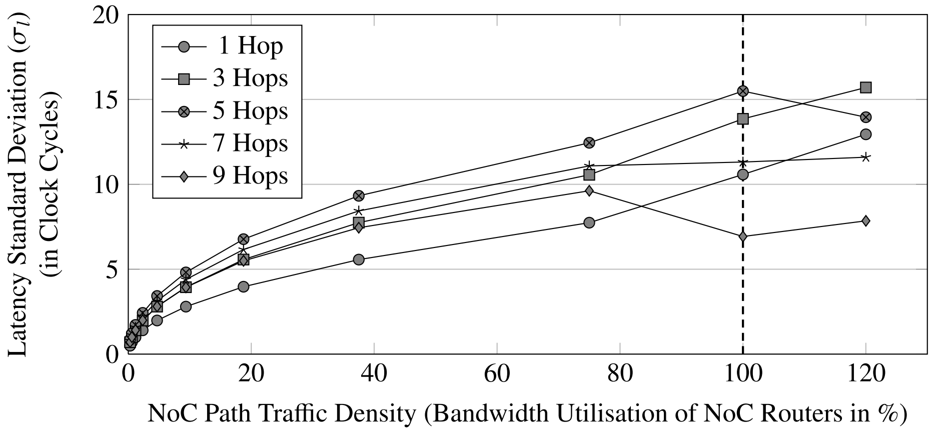
3.1. Latency Jitter in The EMBRACE Mesh Topology NoC Architecture

The reported EMBRACE hardware modular SNN architecture illustrated in figure 1, uses a packet switched NoC for synaptic connectivity between Modular Neural Tiles (MNT) [10] [12]. Routers are connected in North, East, South and West directions, forming a manhattan-style, two dimensional mesh topology NoC architecture. Spike communication between the MNTs is achieved by routing spike information within spike data packets over the network of routers. Each MNT contains a 16:16 fully-connected feed-forward SNN structure as the neural computing module, and interfaces with the attached NoC router for spike communication. Each NoC router transmits spike packets for the 16 output neurons and receives spike packets destined for the 16 input neurons of the attached MNT. The maximum spike rate accepted by the NoC router is one spike packet every 8 clock cycles). The NoC architecture supports unicast packet flow control, where each spike packet contains destination synapse information for a single spike and is routed independently. The NoC architecture uses an XY routing scheme.

The variation in packet transfer latency resulting from the spike communication interference introduced by the EMBRACE NoC architecture has been modelled using EMBRACE-SysC (clock cycle accurate SystemC simulation models [32]). The mean (μ_l) and standard deviation



(a) Mean Latency



(b) Latency Standard Deviation

Figure 2: Mesh Topology NoC Packet Transfer Latency Variations (Mean and standard deviation of latency is measured in terms of clock cycles)

(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. Figure 2 illustrates the packet transfer latency variations (μ_l and σ_l) measured for the multiple hop NoC paths under various traffic density conditions (i.e. bandwidth utilisation values of intermediate NoC routers). The mean latency plot (figure 2a) shows slow variations in the packet latency, while the latency standard deviation plot (figure 2b) shows 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 start to drop the 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

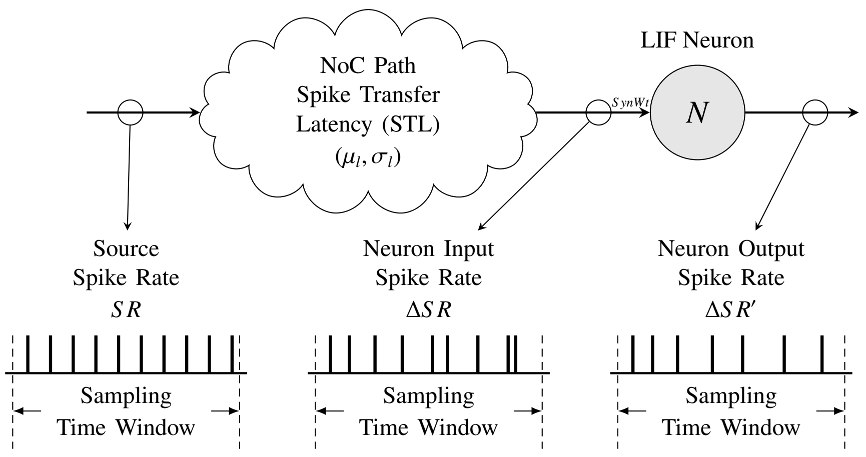


Figure 3: Information Distortion in LIF Neuron Due to Spike Transfer Latency Variations

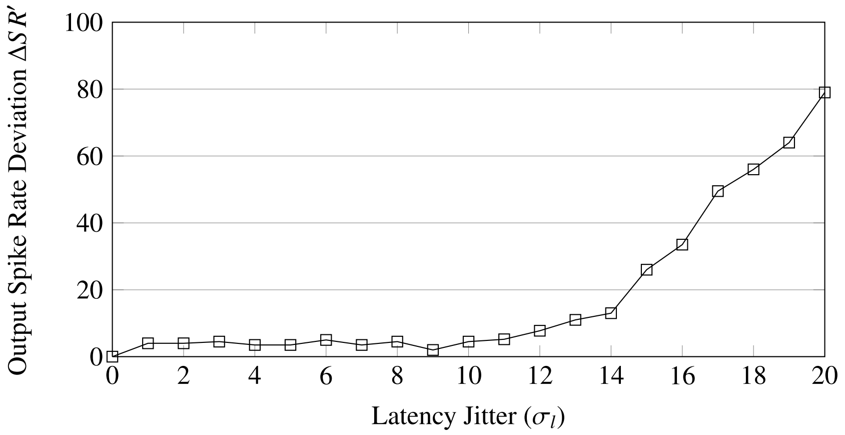
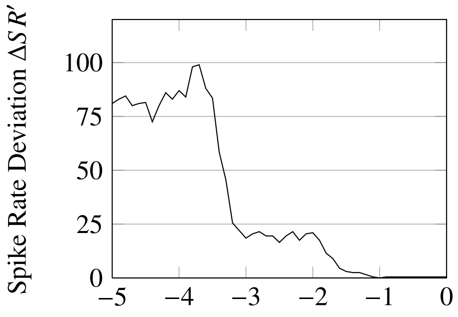


Figure 4: LIF Neuron Output Spike Rate Deviation due to Latency Jitter (Spike rate is measured as number of spike in a STW of 1ms)

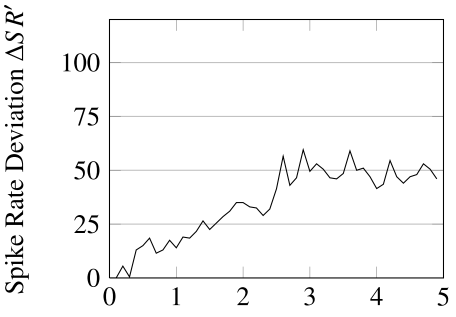
network.

3.2. The Impact of Spike Latency Jitter on SNN Applications

Early research on biological neurons suggests that information in the human brain is contained in the mean firing rate of the neurons (which is in the order of tens of milliseconds) [2] [3] [4]. However, considering the response time of various sensory inputs, another hypothesis suggests that the information may be contained in the precise timing of spike events [5] [6]. Based on these findings, the information encoding used in the SNN computing paradigm can be categorised as *rate based coding* and *temporal coding*. The system inputs/outputs are encoded/decoded as distinct Spike Rates (SR), sampled at fixed time intervals (Sampling Time Window) in the rate based coding technique. Temporal based coding employs precise timing



(a) Inhibitory Synaptic Weight $SynWt < 0$



(b) Excitory Synaptic Weight $SynWt \geq 0$

Figure 5: Spike Rate Deviation (ΔSR) due to Latency Jitter and Synaptic Weight (Constant Latency Jitter $\mu = 15$)(Spike rate is measured as number of spike in a STW of 1ms)

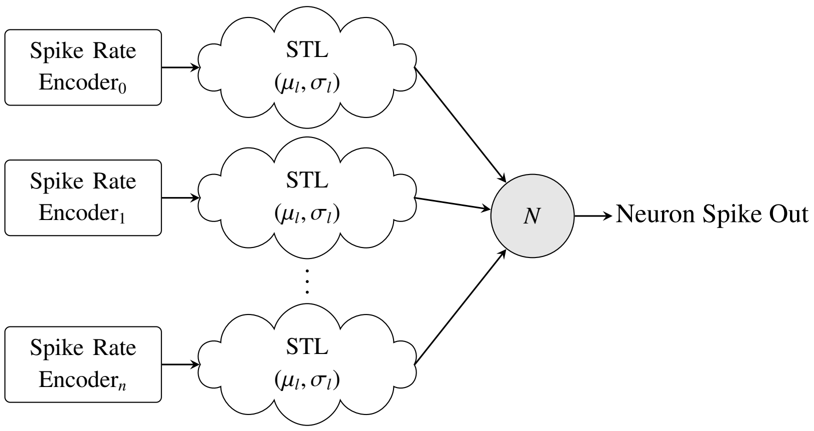
between the spikes, where the system output is decoded based on comparing the occurrence of spikes from multiple SNN outputs.

The packet switched NoC spike communication infrastructure cannot guarantee a precise spike transfer time for a hardware SNN system including a large number (of the order of millions) of virtual synaptic connections. This makes it very difficult to support temporal coding. As depicted in figure 3, the EMBRACE SNN computing model uses spike rate coding for information encoding [9]. This technique offers practical implementation by allowing a sampling time window of a few milliseconds, required by real-life embedded applications. The EMBRACE architecture prototyped on Xilinx FPGA has been successfully applied to benchmark SNN control and classifier applications (such as pole balancer, two-input XOR and Wisconsin cancer dataset classifier) using spike rate coding [9] [11] [12]. The packet switched NoC uses shared buffering and physical channel resources to support a large number of virtual synaptic connections in the EMBRACE architecture. Sharing on-chip communication resources results in variation of the spike packet transfer latency, altering the spike rate and distorting the information in the SNN structure. The remainder of this section analyses the impact of spike packet latency variations on information processed in SNN structures using simulations. The mean (μ_l) and standard deviation (σ_l) values of spike packet transfer latency (l) shown earlier in figure 2 are used for the analysis.

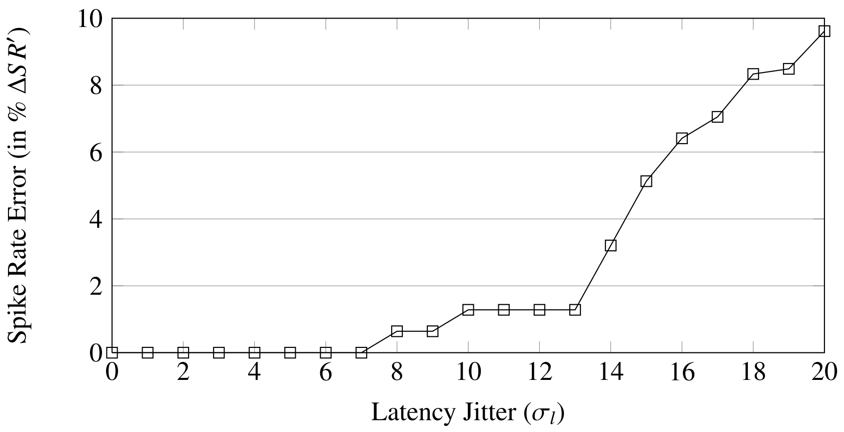
Figure 3 illustrates the distortion of information (which is encoded as fixed spike rate SR), in a Leaky-Integrate-Fire (LIF) neuron [3] due to NoC path spike transfer latency variations. Due to the NoC path Spike packet Transfer Latency (STL) variation (μ_l, σ_l), the source spike rate SR deviates to ΔSR at the input of the LIF neuron. The altered input spike rate translates into the LIF neuron membrane potential variation, resulting in additional error in the neuron output spike rate $\Delta SR'$.

Figure 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 weight synapse, alters the neuron membrane potential significantly and thus has a larger effect on the output spike rate changes due to jitter. Both excitory (positive) and inhibitory (negative) synaptic weight values affect the output spike rate of a LIF neuron due to input spike rate variations.

Figure 5 shows the comparison of LIF neuron output spike rate deviation (ΔSR) due to input



(a) SystemC Simulation Setup for LIF Neuron Multiple Synaptic Inputs Jitter Measurement

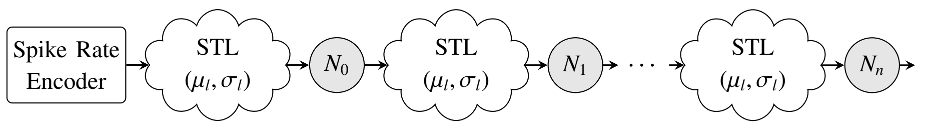


(b) Spike Error Rate for LIF Neuron Multiple Synaptic Inputs (μ_l = 30)

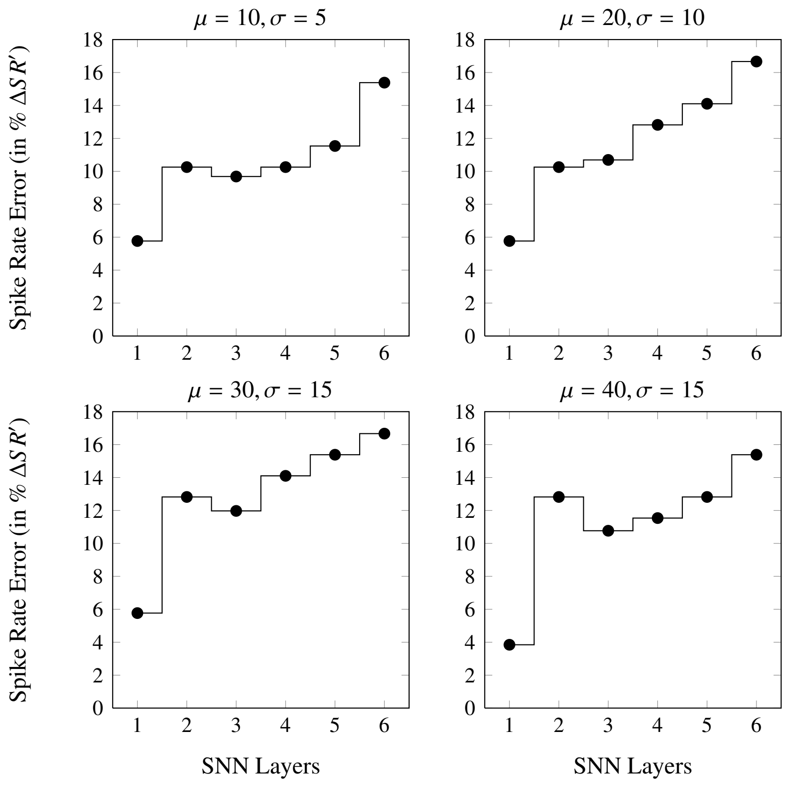
Figure 6: LIF Neuron Multiple Synaptic Paths Jitter (Spike rate is measured as number of spike in a STW of 1ms)

spike rate jitter and excitory/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 in the output spike rate due to input spike rate jitter.

Each synaptic input experiencing spike rate jitter contributes to the output spike rate deviation, according to the associated synaptic weight. Multiple synaptic inputs of a LIF neuron have a combined effect on the output spike rate. Figure 6a shows the SystemC simulation setup to measure the output spike rate error of a LIF neuron due to multiple synaptic inputs with associated spike transfer latency jitter. Figure 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



(a) SystemC Simulation Setup for Spike Rate Error Propagation in SNN Layers Measurement



(b) Spike Error Rate in SNN Layers

Figure 7: Spike Rate Error Propagation in SNN Layers (Spike rate is measured as number of spike in a STW of 1ms)

in membrane potential. As the jitter increases, the changes in membrane potential are significant and have large impact on output spike rate.

SNN application topologies consists of multiple feed-forward and feed-back (recurrent) neuron layers. Figure 7a shows the SystemC simulation setup to measure spike rate error due to cascaded layers in a feed-forward SNN topology. Figure 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 canceled by low spike transfer latency in the next layer. This can be seen as a slight reduction in the spike rate error in third layer in figure 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 figure 4, 6 and 7 shows the traffic dependent behaviour of SNN structure in 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 the SNN application spike rate encoding requirement is 1024 spikes in a STW of 1ms, the maximum spike injection rate is one spike in 196 clock cycles for the system clock frequency of 200Mhz [9] [12]. Also, increase in 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 presents the proposed ring topology interconnect architecture for hardware SNN implementations. Spike packet flow control of the proposed NoC architecture, supporting fixed spike transfer latency is discussed. The section also presents in detail the 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:

- deterministic packet transmission and reception scheduling in the source and destination routers respectively
- a connection topology offering fixed packet transfer latency between the source-destination nodes [30]

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.

Fixed latency flow control of the proposed ring topology interconnect uses timestamping of the input spikes at the source router, and broadcasting of spike packets to all routers within the ring. Each destination router, sorts and buffers spike events based on the received spike packet timestamp value. Buffered spike events are converted to output spikes at the precise clock cycle (corresponding to the source timestamp) determined by the timeslot counter.

Figure 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 Modular Neural Tiles (MNTs) interface with ring routers for spike communication. Each router buffers the spike events generated by the attached MNT and encodes them in spike packets.

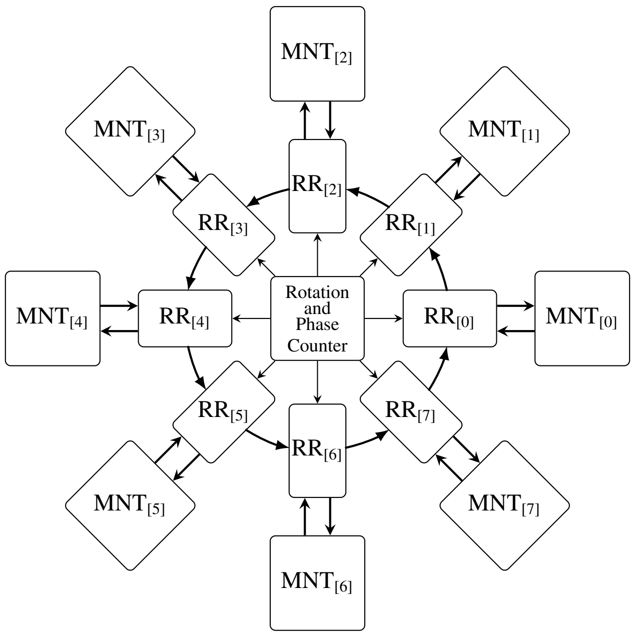


Figure 8: Ring Topology Interconnect Architecture (Number of Ring Nodes R = 8)

Rotation Count $RCount = 6$	Rotation Count $RCount = 7$	Rotation Count $RCount = 0$	Rotation Count $RCount = 1$	Rotation Count $RCount = 2$
Ring Phase $RPhase = FP$	Ring Phase $RPhase = FP$	Ring Phase $RPhase = IP$	Ring Phase $RPhase = FP$	Ring Phase $RPhase = FP$
CLOCK$_{n-2}$ $n \neq \mathbb{N}R$	CLOCK$_{n-1}$ $n \neq \mathbb{N}R$	CLOCK$_{n}$ $n = \mathbb{N}R$	CLOCK$_{n+1}$ $n \neq \mathbb{N}R$	CLOCK$_{n+2}$ $n \neq \mathbb{N}R$

Figure 9: Rotation Count and Insert (*IP*)/Forward (*FP*) Phases of the Ring (Number of Ring Nodes R = 8)

Generated spike packets are processed and forwarded 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 nodes and arrive at the source node) is equal to the number of nodes in the ring ($RC = R$).

The ring interconnect operates in two phases; namely the *Insert Phase (IP)* and *Forward Phase (FP)*. Figure 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 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

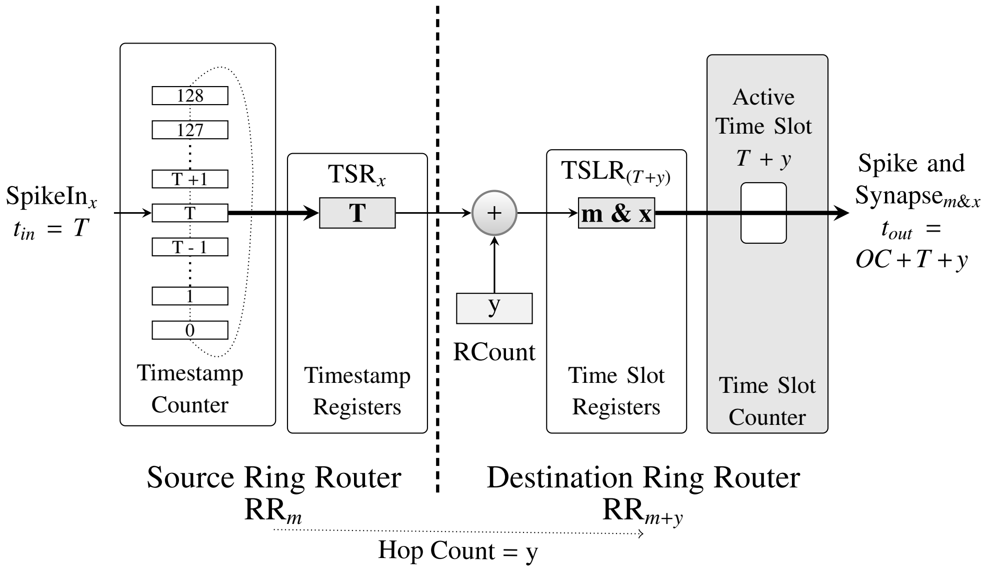


Figure 10: Fixed Latency Spike Packet Flow Control

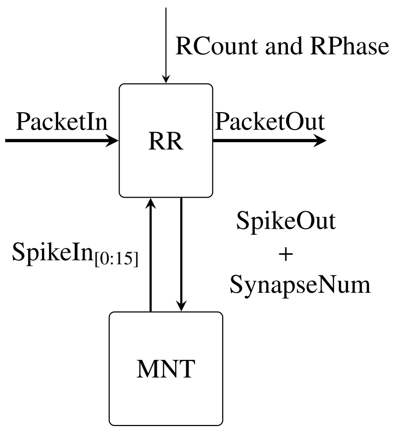
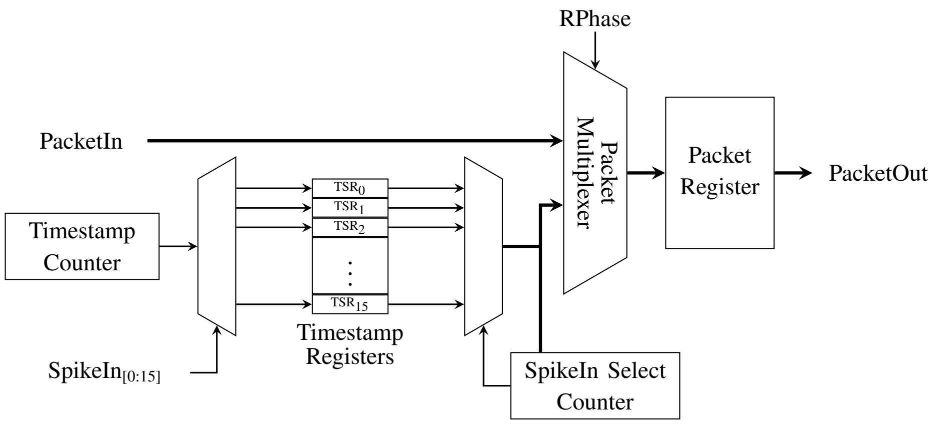


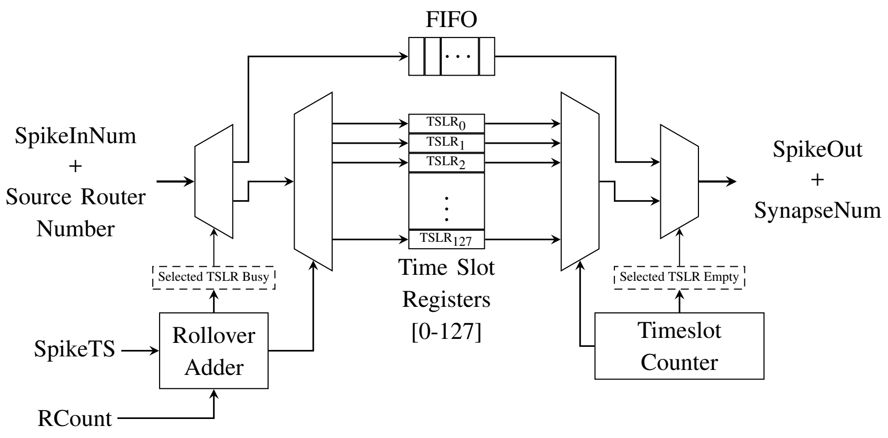
Figure 11: Ring Router and MNT Interface

the phases, the packet received by the router from the previous router is processed to retrieve the spike information. During the insert phase, the router outputs a new spike packet to the next router in the ring. During the forward phase, the received packet is forwarded to the next router in the ring.

Figure 10 illustrates the spike flow control on the proposed ring topology interconnect. The ring router uses a round-robin scheduling policy for transmitting spike packets corresponding to spike inputs (SpikeIn$_{[0:15]}$). A new spike packet is generated in each insert phase. To serve 16 spike inputs, the ring router requires $16RC$ 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



(a) Packet Sender



(b) Packet Receiver

Figure 12: Ring Router Micro-Architecture Organisation

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)

Figure 13: Ring Interconnect Packet Structure

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 forwarded to the next router.

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 node pair. The spike event is then buffered in the Time Slot Registers (TSLR) on the resulting timestamp value. A timeslot counter counting 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 ($16RC$ 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 $ISI \geq OC$.

4.2. Ring Router Micro-Architecture Organisation

The ring routers facilitate spike communication between the MNTs in the ring. Figure 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. Figure 12 illustrates the micro-architecture organisation of the proposed ring router RTL design, comprising (a) *Packet Sender* and (b) *Packet Receiver* subsystems.

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 ($\text{SpikeIn}_{[0:15]}$). 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. Figure 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 *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 operating cycles 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*).

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	Full Ring Rotation 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 consecutive spikes and the spike transfer latency is measured in clock cycles.

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. The performance of the ring architecture has been measured using EMBRACE-SysC design exploration framework for hardware SNN architectures [32]. This section presents spike transfer latency and hardware implementation results for the proposed ring interconnect. Limitations on scalability of the proposed interconnect architecture are also discussed.

5.1. Ring Topology Interconnect Spike Transfer Latency Performance

The performance of the proposed ring topology interconnect has been evaluated using (worst case) spike traffic observed in SNN applications. The measurement setup uses spike rate encoders (instead of MNTs as shown in figure 8) feeding constant spike frequencies to each router in the ring. Each spike rate encoder has 16 spike outputs connected to 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. Table 1 illustrates mean spike transfer latency (μ_l) for multiple hop spike traffic on the proposed ring topology interconnect in an $R = 8$ node configuration. The interconnect offers fixed latency for $ISI \geq OC$. 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 a loss of spikes. Figure 14 illustrates increase in spike loss due to increase in input spike rate. Also, as the spike rate exceeds the upper limit ($ISI < OC$), the probability of received spike events for the destination

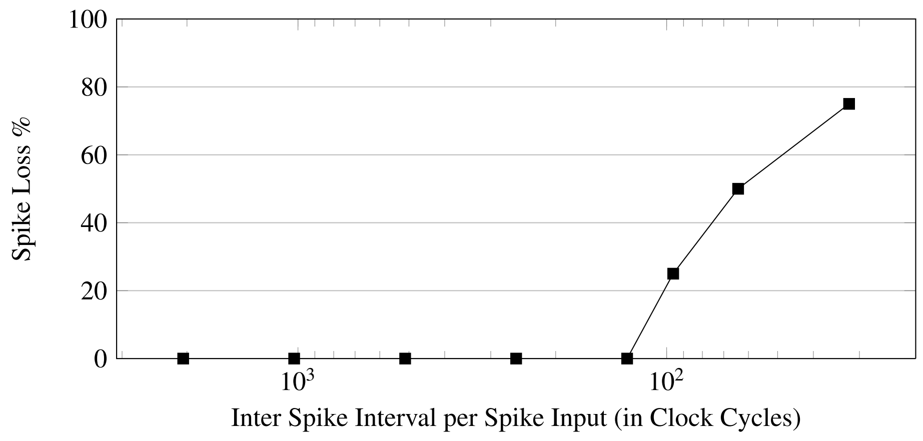


Figure 14: Ring Topology Interconnect Spike Loss Rate

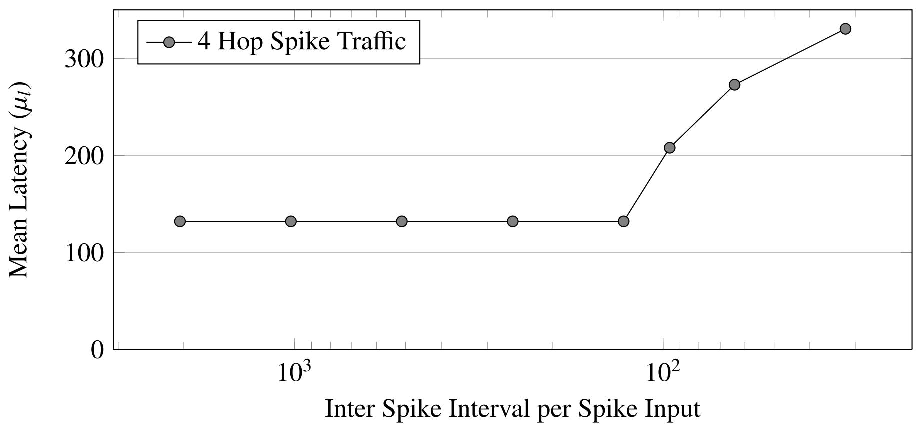
Table 2: Neuron Density, Spike Injection Rate and Maximum Number of Spikes in a Sampling Time Window (STW) of 1ms, for Various Ring Sizes

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

time slot increases, resulting in collisions on destination TSLRs. The packet receiver subsystem within destination ring router, buffers these spike events (causing collisions on the TSLR) in FIFO. Due to unavailability of time slots and untimed delivery of spike events from the FIFO, the spike transfer latency variations increase considerably. Figure 15 illustrates the spike 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 ($OC = R \times SpikeIn_n$) of the ring interconnect imposes the limit on maximum spike rate supported (for spike inputs), for the fixed spike transfer latency operation of the proposed interconnect. The maximum number of spikes for a sampling time window[2] of 1ms for various ring sizes is depicted in table 2. A large number of spikes within a STW allows higher resolution for spike rate encoding in a SNN application setup (figure 3).

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



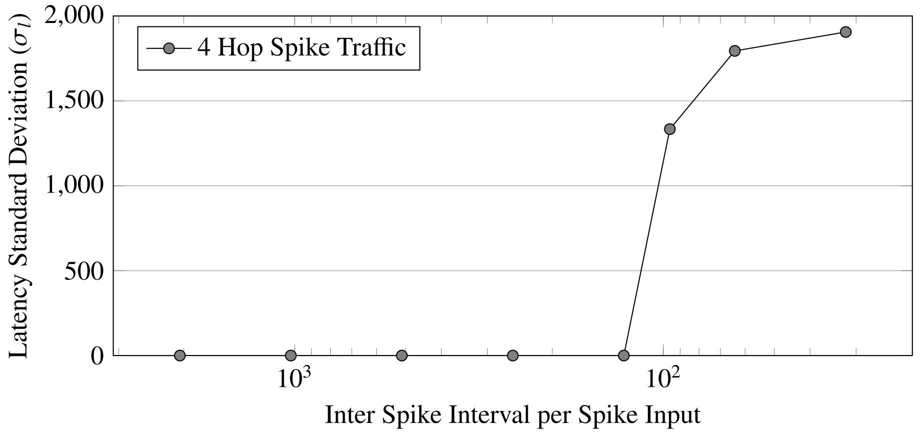


Figure 15: Ring Topology Interconnect Spike Transfer Latency Variations (Inter spike interval is specified in clock cycles between consecutive spikes. Mean and standard deviation of latency is measured in terms of clock cycles)

5.2. Hardware Implementation

The proposed ring topology has been implemented as a configurable design in VHDL. A numbe of ring configurations have been synthesized to Xilinx Virtex 6 FPGA and 65nm low-power CMOS technology from STMicroelectronics. Table 3 shows the hardware synthesis results for the proposed ring topology interconnect. ASIC synthesis has been performed using Synopsys Design Compiler 2009-sp5 and FPGA synthesis using Xilinx XST 13.2.

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 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 number of synapses in the ring. As the ring dimension scales up, 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, which results in a proportional

Table 3: Ring Topology Interconnect Synthesis Results
(Clock Frequency: 200Mhz)

	ASIC Synthesis (65nm Low-Power CMOS)		FPGA Synthesis (Xilinx Virtex-6 FPGA)		
Ring Size	Ring NoC Area (in μm^2)	Ring Router Area (in μm^2)	Ring NoC Slices	Ring Router Slices	Ring Router Slice Registers
4	40580	10136	1060	265	581
8	**157700**	**19707**	**5202**	**650**	**1175**
16	642555	40145	24010	1212	2486
32	2723136	85099	NA	2970	5322

increase in slice register count and associated control logic mapped on the slices.

The previously reported EMBRACE mesh topology NoC uses unicast packet flow control for the synaptic connections between MNTs. The architecture uses SNN topology memory for storing the spike packet information for the large number of synaptic connections. The topology memory within each MNT occupies 50% silicon area [12]. 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 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 figure 8.

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] [17] [18] [19] [20] [21] [22] [23] aim 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 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*.

Research shows that real world biological networks exhibit high communication locality [33] [34]. The modular organisation in human brain has been the motivation for the MNN design strategy, which suggests partitioning of application tasks into a number of subtasks [35] [36] [37]. Based on these factors, the proposed interconnect architecture can be scaled by replicating individual Ring Tiles in mesh topology NoC architecture. The individual ring tile is made up of the ring topology interconnect of 7 MNTs and 1 interface tile. Figure 16 illustrates the architecture of the proposed hierarchical modular hardware SNN. The proposed hierarchical, modular hardware

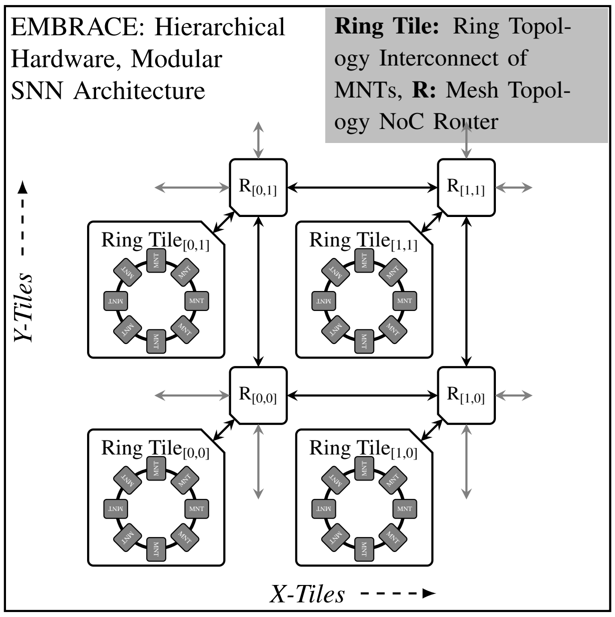


Figure 16: Hierarchical Modular SNN Architecture Comprising Ring Topology Interconnects within Mesh Topology NoC

Table 4: Hierarchical Modular SNN Architecture FPGA Synthesis Results

RTL Entity	Slices	Slice Registers	DSP48E1	BRAM
Neuron	7	16	1	0
Modular Neural Tile	262	1705	32	0
Ring Router	650	1175	0	0
8 Node Ring Interconnect	5202	9408	0	0
Ring Tile	7102	21498	224	1
NoC Router	133	35	0	0
4 Ring Tile Configuration (Figure 16)	30672	98323	672	6

Due to limited FPGA resources, 672 neurons use DSP48E1 slices each, whereas the remaining 224 neurons require 12 slices each.

SNN in a 4 ring tile configuration comprises 896 neurons, 72K synapses and has been synthesized on Xilinx Virtex-6 XC6VLX240T FPGA device. Table 4 illustrates the FPGA resource utilisation for the architecture depicted in figure 16.

6. Conclusions and Future Work

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. Analysis of the effects of NoC latency jitter on the information in SNN structures has been presented. The LIF neuron output spike rate is dependent on the input spike rate, synaptic weight and associated latency jitter on the input synaptic path. Our 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 excitory 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 ($\sigma_l > 13$) 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 with a fixed spike transfer latency flow control has been presented. The interconnect offers a fixed spike transfer latency, which 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 $ISI \geq 128$ clock cycles. 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. The proposed modular hardware SNN comprising 896 neurons and 72K synapses utilises 30672 Virtex-6 FPGA slices.

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 will be developed. 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] 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 10 (9) (1997) 1659 – 1671, ISSN 0893-6080.

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

[4] 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.

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

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

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

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

[9] 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: Field-Programmable Technology, 2009. FPT 2009. International Conference on, 300 –303, 2009.

[10] 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, Int. J. Reconfig. Comput. 2009 (2009) 2:1–2:13, ISSN 1687-7195.

[11] 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 12 (2011) 257–280, ISSN 1389-2576.

[12] 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 .

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

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

[15] 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, 191 – 194, 2004.

[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: Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on, ISSN 1098-7576, 431 –438, 2008.

[18] S. Furber, A. Brown, Biologically-Inspired Massively-Parallel Architectures - Computing Beyond a Million Processors, in: Application of Concurrency to System Design, 2009. ACSD '09. Ninth International Conference on, ISSN 1550-4808, 3 –12, 2009.

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

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

[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, Neural Networks, IEEE Transactions on 18 (5) (2007) 1472 –1487, ISSN 1045-9227.

[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, vol. 3512 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, ISBN 978-3-540-26208-4, 1–24, 2005.

[23] R. Vogelstein, U. Mallik, J. Vogelstein, G. Cauwenberghs, Dynamically Reconfigurable Silicon Array of Spiking Neurons With Conductance-Based Synapses, Neural Networks, IEEE Transactions on 18 (1) (2007) 253 –265, ISSN 1045-9227.

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

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

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

[27] 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, 8 pp., 2003.

[28] 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, ISBN 978-1-4244-4142-6, 144–152, 2009.

[29] 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, ISBN 978-1-60558-774-5, 5–10, 2009.

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

[31] S. Carrillo, J. Harkin, L. McDaid, S. Pande, S. Cawley, B. McGinley, F. Morgan, Hierarchical Network-on-Chip and Traffic Compression for Spiking Neural Network Implementations, Networks-on-Chip, International Symposium on 0 (2012) 83–90.

[32] 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: System on Chip (SoC), 2010 International Symposium on, 139 –145, 2010.

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

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

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

[36] G. Auda, M. S. Kamel, Modular Neural Networks A Survey, International Journal of Neural Systems 9 (2) (1999) 129–151.

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