AER-SRT: Scalable Spike Distribution by Means of Synchronous Serial Ring Topology Address Event Representation

Taho Dorta[a], Mireya Zapata[a], Jordi Madrenas[a], Giovanny Sánchez[b]

[a] *Universitat Politècnica de Catalunya, Dept. of Electronics Engineering, Jordi Girona, 1-3, edif.C4, 08034 Barcelona, Catalunya, Spain*
[b] *Inst. Politécnico Nacional ESIME Culhuacan, Of. No 24, Av. Santana N 1000, Delegación Coyoacan, 04260 Distrito Federal, México*

Abstract

Given the massive number of interconnects in Spiking Neural Networks (SNNs), distributing spikes effciently becomes a critical issue for the efficient hardware emulation of large-scale SNNs. In this work, the AER-SRT (Address Event Representation over Synchronous Serial Ring Topology) architecture for spike transmission is proposed. AER-SRT is a light, easily scalable, packet-based solution implemented with high-speed serial link for multi-chip SNN communication. The channel uses a unidirectional, point-to-point connection between nodes, which provides a high transmission speed. Events (spikes) are distributed among all the nodes in a ring-topology pipeline fashion and the synchronous AER guarantees a collision-free scheme. The fast speed and efficient channel usage limits the spike distribution time to values that allow real-time operation for network sizes that can be calculated with simple design equations. Also, in the proposed communication protocol there is no specific or master node, so new nodes can be added to the ring by simply modifying two configuration parameters. As a proof of concept, a prototype of the architecture has been implemented and tested on FPGA development boards.

Keywords: AER (Address Event Representation); multi-chip communication; synchronous serial ring; Aurora protocol; SNN emulation; time slot emulation.

1. Introduction

Spiking Neural Networks (SNNs) are third-generation networks of biological neuron models. They exhibit higher computation capability and they are closer to biological neurons than other artificial models [1]. Recently, the interest in modeling the brain at detail scale and emulating SNNs has significantly grown as research groups and companies have envisaged promising features that systems based on artificial neural models, in general, and SNNs in particular, can offer to develop smart and intelligent systems; and even in longer term to mimic parts of the human brain. Since the estimated number of neurons in the human brain is

around tens of billions and the number of synapses even four orders of magnitude above [2], nowadays only much simpler networks can be simulated or emulated. Understanding by emulation the real-time simulation of the network, if SNN emulation has to be performed by a robot or embedded system, it becomes clear that low-power custom hardware systems must be applied, instead of high-performance but bulky and power-eager computers. As in biological neural networks, very large number of neurons and synapses need to be simulated to reproduce interesting spiking neuromorphic behaviors. Thus, when building SNN hardware, two major challenges arise:

1. Large quantity of neurons and synapses have to be efficiently emulated in parallel or with a balanced combination of parallel/serial processing. Several optimized hardware architectures that address the SNN emulation challenge have been proposed, with different flavors [3, 4, 5, 6].
2. High spiking activity has to be routed or broadcasted to the synapses of the destination neurons in a very short time. The key point is that full connection becomes intractable as the number of neurons grow. This work concentrates on this second challenge.

Neuron spikes are produced at low rates, normally around hundreds of Hertz [1], and typically they have to be transmitted to thousands of receiving synapses. Thus, instead of costly dedicated connection lines, a successful spike time-multiplexing bus, using Address Event Representation (AER), was proposed in the early nineties [7]. AER encoding is based on the binary model of the spike, so it is not necessary to transmit the spike but only the spiking neuron address through a single, parallel broadcast bus, which optimizes the resources required for spike distribution. Receiving chips or subsystems decode the address and drive the spike to the corresponding synapses. This approach, originally asynchronous, has a major limitation when several neurons spike at the same time. In that case, spikes may be lost or at least their timing is degradated. Furthermore, complex arbitration logic is required in the latter case.

In order to circumvent these issues, time-slot emulation [8] and synchronous AER bus [9] were proposed. In this case, spike events are assigned to time slots and time multiplexing distribution is applied. Spike time resolution is limited by the time slot width, but no spikes are lost. The controlled time uncertainty can be reduced as the time slot is made small.

The spike communication scheme is fundamental to distribute the maximum number of spikes while preserving the time window imposed by the real-time behavior in this kind of systems. The maximum number of neurons and synapses is limited by the spike distribution efficiency, which becomes thus a key aspect to enable the emulation of large-scale neuromorphic systems. Since a limited number of neurons can be emulated in a specific silicon device, the use of multi-chip systems is very common when building hardware SNN. In this kind of systems inter-chip communication becomes a critical point, and usually it is responsible of scalability degree and efficiency of the whole system.

Use of AER protocol has become very popular for chip interfacing in this kind of multi-chip neuromorphic systems. The way AER is implemented has evolved to more efficient, high-speed serial communication. Serial AER utilizes fewer wires, exhibits better performance and does not limit the data word-width. Several serial AER systems have been reported in literature demonstrating better performance than traditional parallel bus [10, 11, 12, 13].

In [6] a packet-based synchronous AER protocol is presented. It is based on multilayer wafer scale communication architecture with very high bandwidth. The same channel is used for distributing events and for configuration. In [13] different topologies for AER systems are studied at theoretical level. Here an interesting 2D mesh topology is presented, where an AER router is used in each node to redirect traffic. This approach is similar to solutions used in NoC (Network on Chip) [14].

Our solution presented here, AER-SRT, allows very simple SNN chip interconnection by means of a ring topology network. It balances between scalability and efficiency, achieved using point to point high-speed differential serial transceivers to do inter-chip communication, at frequencies of several Gbps. This high speed and throughput, largely compensates the latency introduced by the pipeline. A tradeoff between scalability and efficiency is obtained, which results in a very interesting solution to build this kind of systems.

In the following section, the details of AER-SRT protocol are presented. In Section 3, details about hardware implementation of an AER-SRT network interface are given. Experimental results on a FPGA prototype are shown and discussed in Section 4. Finally, in Section 5 the conclusions of the work are pointed out.

2. AER-SRT: A Synchronous Serial AER Protocol

AER-SRT is a synchronous packet-based protocol that allows efficient distribution of AER events in a ring-topology multi-chip Spiking Neural Network. AER-SRT encapsulates traffic using Xilinx Aurora 8B/10B, an open IP core that implements a light high-speed serial protocol, being is especially indicated for multi-chip communication [15]. As seen in Fig 1, connection between nodes is point-to-point and unidirectional.

AER-SRT uses two configuration parameters in each node: The Ring Size, to define the number of nodes in the system, and a 7-bit Chip Identifier (Chip Id) for detecting the source of each transmitted event, which allows a scalability of up to 128 nodes. Modification in the number of nodes is straightforward and it only implies to reconfigure Ring Size parameter at each node. The protocol remains the same regardless of the node count, since there is no master or special node.

To test this protocol a SNN emulator has been implemented, which generates a succession of emulation cycles. Every emulation cycle is divided in two phases:

1. **Execution Phase (EP).** During this phase synapses and neural algorithms are calculated. The generated events are later distributed through the ring.

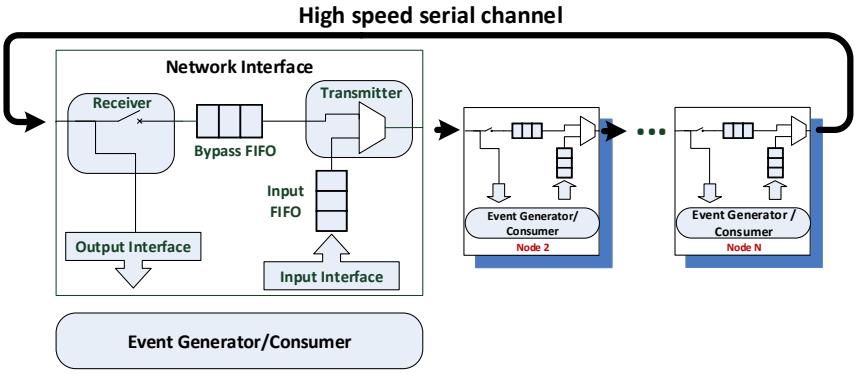

Figure 1: AER-SRT Ring

2. **Distribution Phase (DP).**
   (a) *Ring Synchronization Phase (RSP)*: Since every node in the system does not necessarily finish the execution phase at the same time, a packet-based synchronization mechanism has been implemented. In this stage each node waits until the remaining are ready to start distributing events.
   (b) *Event Transmission Phase (ETP)*: Once the ring is synchronized, every node starts sending events to the next node in a burst. The DP ends when all packets reach every chip.

In AER-SRT a 16-bit packet length is defined. The most significant bit of every packet defines a data or control packet, set to 1 or 0 respectively. Data packets encapsulate AER events and Control packets are specific packets to support the transmission protocol. A 3-bit header currently allows to distinguish between 4 types of control packets: SYNC, START, FINISH and IDLE.

SYNC packets are used in RSP. Each node sends this control packet to indicate it is ready to start distributing events and after that it propagates the incoming SYNC packets from the other nodes. As the number of received SYNC packets in the node equals the Ring Size parameter, the ring will then be synchronized and the distribution of events (ETP) starts.

The START packet contains the Chip ID of each node. Once the ETP initiates, a START packet is sent leading the block of events belonging to each node, every node sends all data sequentially. Thanks to this fact, the overhead is reduced by sending the START packet only once, before all the followings data packets of a given node

The packets travel throughout the ring until they eventually return to the same node where they were generated. Thus, every received START package is compared with the Chip Id. If the identifier is the same, then the associated event data have travelled all the ring and they are removed. Otherwise, the associated packets have to be forwarded to the next node and continue travelling along the ring.

An added benefit of the ring architecture is that each node can compare the events that have circulated the ring with the events it transmitted, being able to confirm the information integrity or to detect any transmission error.

Every node in the ring sends one FINISH packet after it has sent the last data packet. This way, when a given node receives a FINISH packet from another node, no more events will be transmitted from that node in the current time slot. When FINISH packets from all nodes have been received, the DP is over.

It is important to highlight that each node is in charge of removing the events that it has generated.

During the EP events are not distributed. In order to keep the high-speed serial link up and not to lose synchronization, the AER-SRT network interface sends IDLE packets.

Error Control

The proposed protocol and ring topology easily permits without cost ensure full data integrity or to detect any transmission error, because each node will receive all the packets generated by itself under correct operation. In the implemented AER-SRT solution, error detection consists in counting the number of Input Buffer data packets and compare it with the number of received data packets with the node Chip Id. In order to support error correction, the protocol could be modified, as future work.

3. AER-SRT Ring Architecture

In this section a hardware implementation of an AER-SRT ring is presented. Based on the block diagram of Fig. 1 an AER-SRT implementation is described. For simplicity, every SNN node is considered to consist of two blocks:

3.1. Event Generator/Consumer

For the purpose of debugging and benchmarking, this module represents the neuron and synapse emulating array that generates and consumes spikes. It can be configured to produce different traffic load independently for each node.

3.2. Network Interface

The main Network Interface control signals are: AER_eo_exec, AER_ON, AER_done and AER_eo_distrib. These four signals are responsible of delimiting different phases in the protocol. AER_eo_exec is a signal from Input Interface that indicates EP is over and RSP can start. AER_ON indicates RSP is complete and ETP can start. AER_done and AER_eo_distrib indicate DP is over, and consequently next EP can start.

The Network Interface in detail is introduced in the block diagram Fig. 2.

- The *AURORA CORE* consists of the Transmitter and Receiver Sides
- The *Aurora Transmitter* (TX Side) serializes 16-bit AER-SRT packets and it forwards the traffic serially through the intra-chip high-speed link.

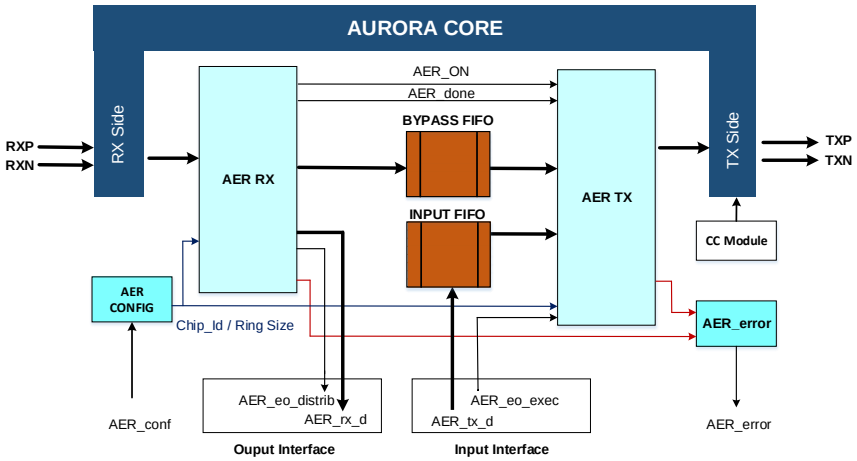

Figure 2: AER-SRT Network interface Diagram Block

- The *Aurora Receiver* (RX Side) de-serializes 16-bit AER-SRT packets from incoming traffic and sends them to AER RX block.
- The *Input Interface* is responsible of receiving the node events from the Event Generator/Consumer that have to be distributed and it indicates when EP is over, so synchronization phase can start.
- The *Output Interface* sends to a node all the events generated by all the nodes in the system when the DP is over.
- As buffers, the Network Interface uses 2 FIFOs of 1024 word depth: *Input FIFO* keeps events produced by the Event Generator/Consumer during EP, until DP starts. *Bypass FIFO* receives the traffic that has to be retransmitted. The FIFO size introduces a limitation in the maximum number of spikes (i.e., emulated neurons) that a simple node can generate and retransmit, but it can be easily adapted to support nodes with higher density of neurons.
- The *AER TX* module is responsible of generating the AER-SRT frames for the Aurora TX Side block from Input FIFO and also retransmit the received packets from Bypass FIFO. This module enables different operations depending on the protocol stage. a) When the EP is over (AER_eo_exec on), it sends SYNC packets and waits for AER_ON signal which indicates RSP is over. Once AER_ON is received, in the following step the buffered events are transmitted. Data transmission is performed in the following sequence: First a START packet is sent, followed by all the data packets containing the events of the Input FIFO and finally a FINISH packet. b) After transmitting its own events, the node enters bypass mode to distribute all the data that are already in the ring. c) Finally, when

AER_done signal is received, which means DP is over, it transmits IDLE packets to keep channel synchronization.

- *AER RX* module extracts AER-SRT packets from Aurora RX Side in order to send them to the Event Generator/Consumer, and it is also responsible of writing the traffic to be retransmitted into the Bypass FIFO, its operation involves:

  a) Checking whether the currently received AER-SRT data block has to be forwarded or not, i.e., whether the packet has been transmitted by another node or by the same node. It compares the START packet Chip Id with the node Chip Id. It writes the data packets into the Bypass FIFO only if the comparison is false. b) Detecting the START packet and extracting the Chip Id in order to identify the source of all the data packets that follow. c) Counting the incoming SYNC packets, comparing the number of received SYNC packets with the Ring Size parameter and generating AER_ON signal when the comparison is true. d) Detecting the finish condition counting the received FINISH packets, comparing this number with Ring Size and generating AER_done signal when the comparison is true. e) Extracting AER events from the data packets, adding the Chip_Id extracted from the START packet to the most significant part of the AER address, and sending the AER events to Output Interface to eventually process them as incoming spikes.

There are also other ancillary modules as Error Detection Block (AER_error) and Configuration Block (AER_CONFIG), responsible of saving the configuration values Chip Id and Ring Size. Finally, the Clock Compensation Module (CC Module) adjusts slight clock frequency differences among ring nodes.

4. Implementation, Experimental Results and Discussion

The AER-SRT system presented in the previous section has been completely described using VHDL. As previously mentioned, a block that simulates the event generation and consumption of the neural emulator has also been defined (Event Generator/Consumer block in Fig. 1). It simulates the handshake signals for monitoring states and it generates events with the format required for the AER-SRT protocol. The node description containing the AER-SRT and the Event Generator and Consumer has been synthesized and implemented on Xilinx FPGA KC705 (Kintex 7) evaluation boards. These boards include SMA connectors that allow the required two-port high-speed connections, TX and RX for the ring implementation.

In particular, the implementation uses high-speed serial GTX transceivers integrated in the FPGA wired to differential lines terminated with SMA connectors. The shared communication channel uses a point-to-point connection between nodes operating at a serial bit rate of 3.125 Gbps. The connection between the transmitter and receiver is implemented using differential transmission lines. For debug and on-board validation, a virtual input/output (VIO),

Logic Utilization	Used	Available	Utilization
Slice Registers (Flip-Flops)	4332	407600	1.06%
Slice LUTs	2008	203800	0.98%
LUT as Logic	1564	203800	0.76%
LUT as Memory	444	203800	0.69%
RAMB36/FIFO	2	445	0.45%

Table 1: Device utilization Xilinx Kintex7 XC7K325T2FFG900C (KC705 evaluation board).

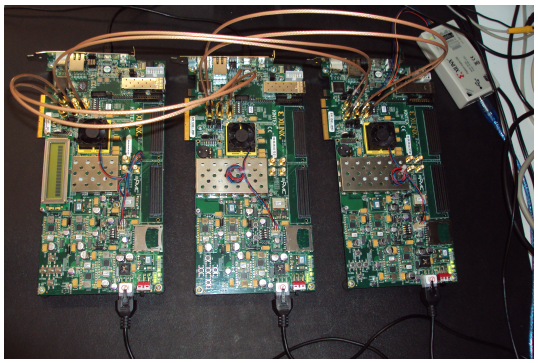

Figure 3: Picture of the 3-node AER-SRT experimental system.

an integrated logic analyzer (ILA) and Xilinx ChipScope Analyzer were used. Previous simulations were performed using QuestaSim. Xilinx Vivado tool was used for synthesis and implementation. Table 1 shows the hardware resource usage on the FPGA for one node. These results illustrate the hardware efficiency showing a very low resource utilization around 1% for the FPGA, leaving the majority of resources for the implementation of the custom neural emulator.

Fig. 3 presents a prototype of 3-node AER-SRT. This configuration is a ring of three Xilinx KC705 FPGA Evaluation Boards. Each FPGA contains a node including the Event Generator and Consumer and the AER-SRT Network Interface. Each node can be configured to send different number of spikes. Each node has a pair of TX lines and a pair of RX lines, as shown in the picture.

Fig. 4 shows a snapshot, obtained with the ChipScope tool, of the AER-SRT ring real-time execution. It illustrates the number of clock cycles utilized to distribute 3000 spikes in the 3-node ring. Each node has been set to generate 1000 spikes at every execution cycle. This configuration is representative since 1000 spikes/node is close to the maximum capacity of each node. As stated before, the node capacity is limited by the size of Input and Bypass Buffers. This setup allows to get the measure of the worst-case time distribution for the 3-node system during high spiking activity.

Looking at signal aer_rx_data_chip_id in the time diagram, it can be seen in the center of the figure how Chip_Ids are received at node 01, starting from node 02, then 03 and finishing with the own 01. As indicated, 1000 events from

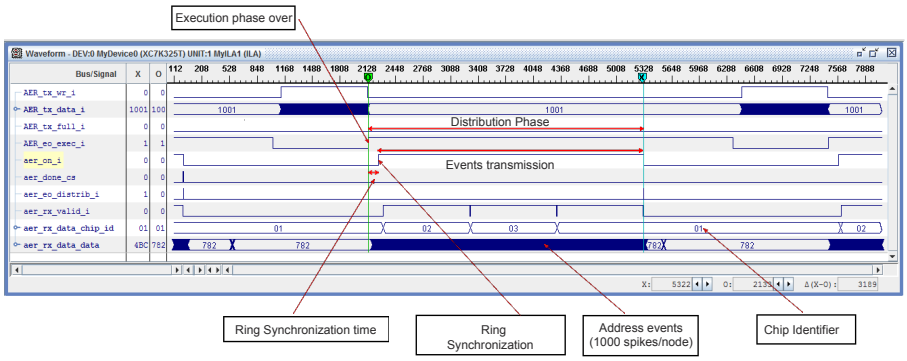

Figure 4: Chipscope snapshot. Distribution of 1000 spikes/node in a ring of 3 AER-SRT nodes

each node are received and processed after the corresponding Chip Id. Signal aer_rx_valid_i indicates the time frame when valid spikes are being received.

In the example figure, the total number of cycles for Distribution Phase (DP) is 3189, requiring 1.063 cycles per event, thus implying a 6.3% total overhead. As described in Section 2, the DP is divided into Ring Synchronization Phase (RSP) and Event Transmission Phase (ETP). The delay needed for RSP corresponds to the delay between AER_eo_exec and AER_ON signals. This synchronization time is approximately proportional to the latency of a single node multiplied by the number of nodes in the ring. Specifically, for the 3-node configuration, 121 cycles are utilized.

During ETP, the number of cycles is equal to the total number of spikes plus a small overhead, in particular, 3068 cycles are used for actual distribution of the spikes. The difference of 68 cycles from the ideal 3000 cycles is due to packets used for monitoring the handshake sequence of the protocol and for the overhead imposed by the Aurora core which automatically interrupts the transmission of data to send clock compensation sequences every 10,000 bytes[15]. This behavior is replicated for bigger systems, as simulation analysis confirms.

Fig. 5 illustrates the scalability of the system showing the RSP and ETP cycles as a function of the number of nodes. The data has been obtained experimentally for rings of 1, 2 and 3 nodes and by simulation for rings with more than 3 nodes. The blue data corresponds to the RSP , which does not depend on the spike number, and the red and green data to rings generating 500 spikes/node and 1000 spikes/node in the ETP, respectively. The interpolating lines of RSP and ETP have been obtained using linear regression, so they approximate the actual values for all points.

The graph clearly show linear dependencies on the number of nodes. The system needs in both cases an average amount of RSP cycles (approximated to 38 cycles/node) proportional only to the number of nodes plus a small offset and it grows much slower than the ETP cycles used for a relatively high spiking

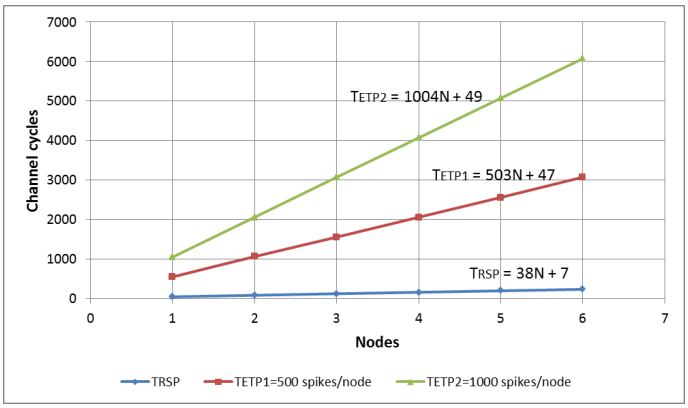

Figure 5: RSP and ETP cycles depending on the number of nodes. a) RSP cycles (blue line). b) ETP cycles for 500 spikes/node (red line) and c) ETP cycles for 1000 spikes/node (green line).

activity. On the other hand, the distribution cycle number is proportional to both the number of nodes and the amount of spiking activity in every node.

From the simulated and experimental data, adding the RSP and ETP contributions, a general expression, shown in Eq. 1, has been extracted. It corresponds to the total time taken for the DP as a function of the number of nodes:

$$t_{dist} = \frac{1}{f_{CLK}} \left(\sum_{n=1}^{N} s_n + 42N + \vartheta \right) \tag{1}$$

where N represents the total number of nodes in the ring; s_n is the number of spikes generated by each node; $f_{CLK} = 125$ MHz corresponds to the clock frequency of this prototype. The first term of the equation accounts for the ideal time, which is the total number of spikes. The second and third terms, $42N + \vartheta$, are proportional to N and offset terms that correspond to overhead due to the RSP and to the control cycles introduced automatically by Aurora core during framing character stripping, left alignment processes and clock compensation. These operations are required to ensure a reliable transmission, keeping both the channel up as well as the synchronization of the communication process and their effect on the total time. Approximately, it can be estimated as $\vartheta = 56$ cycles. Slight variations in slope are also expected from the performed linear regression.

It becomes clear from Fig. 5 and from Eq. 1, and it is important to highlight, that the system complexity scales linearly.

According to references [16, 17], a 1 ms time slot reference for emulating a whole synaptic and neural cycle plus spike transmission in real time is assumed. During this time, the neural and synaptic algorithms need to be executed (Execution Phase, EP) and the outcoming spikes have to be distributed to the destination synapses (Distribution Phase, DP). For simplicity, this time, called

emulation cycle, is equally allocated in two parts: 0.5 ms for the EP and 0.5 ms for the DP. The objective is to distribute the maximum amount of events through the ring using the $\Delta t = 0.5ms$ DP time window.

The size of the AER-SRT input buffer introduces a limitation in the number of spikes each node is able to generate during one emulation cycle. In the reported implementation the size is set to 1000 events. In the tradeoff between number of nodes and number of spikes per node, obviously smaller ring size (fewer nodes) results in better performance and better efficiency, since latency increases with ring size. Taking this into account, the best configuration for distributing spikes during Δt corresponds with nodes that generate the maximum number of spikes. The number of clock cycles in time Δt is $c = \Delta t.f_{CLK} = 62,500$, for $f_{CLK} = 125MHz$ of our prototype. Deducing the calculated 6.3% overhead, the number of cycles corresponds to 58,562 events that can be transmitted in real time.

It is important to note that this maximum event capacity allows full connectivity between all the neurons. It means all spikes can be broadcasted to all the neurons in the system, which is not the case in many other implementations. The bandwidth cost for this full-connectivity potential is the same if at least one single neuron in each node is to be reached; however, for sparser networks, a hierarchical topology could be applied without much design effort, saving the transmission of local spikes and increasing much more the maximum size of the real-time network.

Instead of the measures 58.56 Mevent/s of our prototype, in order to compare with the raw speed considered in other works, if only the transmission phase is considered, the maximum transmission speed is around 117 Mevent/s, same order of magnitude of other previous works [10, 11, 12]; however it is still far from the bandwidth obtained in the FACETS multilayer wafer scale neuromorphic system [18, 6].

The rates achieved in these works are 500 Mevent/s at inter-board level and 1.28 Gevent/s at FPGA-to-FPGA top level. The very high throughput is due to the use of a higher FPGA clock frequency of 500MHz, transceivers at 10 Gbps, using Aurora protocol, and several parallel links. Upgrading our prototype to a 500 MHz clock and 4, 10 Gbps transceivers, our approach would show similar performance. However, with our approach events contention is not possible because specific time slots are used for sending spikes through the shared channel. As a result, a free collision transmission and thus a minimum error rate is obtained. As another example, in designs using ALOHA, such as in [19], access protocol transmits events as soon they are generated, however time overlaps can produce high error rates and limit the throughput for sparse spiking activity.

The Ring Topology presented here is easy to implement and the penalty introduced in latency is only constrained to 42 clock cycles per multi-chip node. More complex topologies as 2D router-based mesh is presented in [13]. Despite this solution is very promising and bandwidth could be theoretically better than ring topology, complexity associated with node configuration and data distribution may insert additional latency and increase of hardware. Moreover,

Multi-FPGA configuration has only been tested for 2 FPGA configuration in that work.

5. Conclusion

In this work, the AER-SRT architecture, which consists in a scalable and efficient solution to synchronously transmit address events for multi-chip spiking neural emulation systems, has been proposed. It is based on a ring topology using serial, high-speed point-to-point connections where each node only needs two communication ports: one for reception and another for transmission. The architecture uses low-complexity hardware and provides reliable data delivery. Based on the time-slot emulation scheme, the synchronization among nodes prevents spike contention or spike loss.

The ring protocol is based on packages to monitor the start and finish of event distribution in all the nodes. Additionally it only requires 2 configuration parameters for attaching a new node to the ring: Ring Size and Chip Id, without incurring in extra hardware overhead. The symmetric design does not require a dedicated hub working as a master. As a consequence of using a serial ring topology, transfer rate is not affected when a new node is connected to the ring. The bus efficiency is neither reduced as the number of nodes connected to it increases. Regarding transmission errors, the architecture supports by construction error detection. Error correction could be implemented by extending the architecture, at the cost of some control overhead.

The AER-SRT architecture has been implemented on FPGA evaluation boards and its operation experimentally demonstrated by real hardware test of a 3-node ring generating 1,000 spikes each node and by precise simulation of rings up to 6 nodes. The AER-SRT delay scales linearly with the total number of spikes to transmit, regardless of the spikes that each node generates, providing an architecture that can be easily upgraded to support larger configurations.

The experimentally achieved throughput is close to 1 spike per clock cycle with a latency of approximately 42 cycles per ring node. This allows to use this implementation to transfer large amounts of events in a free-collision channel while maintaining the temporal window of 0.5 ms which permits real-time operation. Extending the results for bigger systems, the architecture supports building a system able to broadcast approximately 60k events in real time in a 60-node ring system working at 125 MHz. Better performance and bigger capacity is possible increasing the clock frequency and link speed.

In summary, the AER-SRT architecture allows efficient broadcast event distribution in a multi-chip configuration, with point-to-point connections and ring topology, offering a good choice for integration in Spiking Neural Network (SNN) real-time emulators.

Acknowledgement

This work was supported in part by the Spanish Ministry of Science and Innovation under Project TEC2011-27047, and the European Social Fund (ESF).

Mireya Zapata holds an scholarship from National Secretary of High Education, Science, Technology, and Innovation (SENESCYT) of the Ecuadorian government.

6. References

[1] M. Wolfgang, Networks of spiking neurons: The third generation of neural network models, Neural Networks 10 (9) (1997) 1659–1671. doi:10.1016/S0893-6080(97)00011-7.

[2] R. Cattell, A. Parker, Challenges for Brain Emulation : Why is Building a Brain so Difficult ? (2012) 1–28.

[3] S. Furber, F. Galluppi, S. Temple, L. Plana., The SpiNNaker Project, Proceedings of the IEEE (2014) 1–14doi:10.1109/JPROC.2014.2304638.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6750072

[4] S. Moore, P. Fox, S. Marsh, A. Markettos, A. Mujumdar, Bluehive - A Field-Programable Custom Computing Machine for Extreme-Scale Real-Time Neural Network Simulation, 2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines (2012) 133–140doi:10.1109/FCCM.2012.32.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6239804

[5] R. E. Ros, E. Ortigosa, Agís, R. Carrillo, M. Arnold, Real-time computing platform for spiking neurons (RT-spike)., IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council 17 (4) (2006) 1050–63. doi:10.1109/TNN.2006.875980.
URL http://www.ncbi.nlm.nih.gov/pubmed/16856666

[6] S. Scholze, H. Eisenreich, S. Hoppner, G. Ellguth, S. Henker, M. Ander, S. Hanzsche, J. Partzsch, C. Mayr, R. Schuffny, A 32gbit/s communication soc for a waferscale neuromorphic system, Integration, the VLSI Journal 45 (1) (2012) 61 – 75. doi:http://dx.doi.org/10.1016/j.vlsi.2011.05.003.
URL http://www.sciencedirect.com/science/article/pii/S0167926011000538

[7] M. Mahowald, Vlsi analogs of neuronal visual processing: a synthesis of form and function, Ph.D. thesis, California Institute of Technology (1992).

[8] J. Madrenas, J. Moreno, Strategies in SIMD Computing for Complex Neural Bioinspired Applications, in: Adaptive Hardware and Systems, 2009. AHS 2009. NASA/ESA Conference on, IEEE, 2009, pp. 376–381.
URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5325429

[9] J. Moreno, J. Madrenas, L. Kotynia, Synchronous Digital Implementation of the AER Communication Scheme for Emulating Large-Scale Spiking Neural Networks Models, in: Adaptive Hardware and Systems, 2009. AHS 2009. NASA/ESA Conference on, IEEE, 2009, pp. 189–196.
URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5325453

[10] H. Berge, P. Hafliger, High-speed serial aer on fpga, in: Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on, 2007, pp. 857–860. doi:10.1109/ISCAS.2007.378041.

[11] D. Fasnacht, A. Whatley, G. Indiveri, A serial communication infrastructure for multi-chip address event systems, in: Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on, IEEE, 2008, pp. 648–651.

[12] T. Iakymchuk, A. Rosado, T. Serrano-Gotarredona, B. Linares-Barranco, A. Jimenez-Fernandez, A. Linares-Barranco, G. Jimenez-Moreno, An aer handshake-less modular infrastructure pcb with x8 2.5gbps lvds serial links, in: Circuits and Systems (ISCAS), 2014 IEEE International Symposium on, 2014, pp. 1556–1559. doi:10.1109/ISCAS.2014.6865445.

[13] C. Zamarreno-Ramos, A. Linares-Barranco, a. B. L.-B. T. Serrano-Gotarredona, Multicasting mesh aer: A scalable assembly approach for reconfigurable neuromorphic structured aer systems. application to convnets, Biomedical Circuits and Systems, IEEE Transactions on 7 (1) (2013) 82–102. doi:10.1109/TBCAS.2012.2195725.

[14] F. Morgan, S.Cawley, J. Harkin, L. M. D. B. Mc Ginley, S. Pande, An Evolvable NoC-Based Spiking Neural Network Architecture, Neuron.

[15] Xilinx, Aurora 8B/10B Protocol Specification, http://www.xilinx.com/support/documentation/ip_documentation/aurora_8b10b_protocol_spec_sp002.pdf/ (2014).

[16] J. Iglesias, A. Villa, Effect of stimulus-driven pruning on the detection of spatiotemporal patterns of activity in large neural networks, Biosystems 89 (1-3) (2007) 287–293, selected Papers presented at the 6th International Workshop on Neural Coding Neural Coding 2005 6th International Workshop on Neural Coding. doi:http://dx.doi.org/10.1016/j.biosystems.2006.05.020.
URL http://www.sciencedirect.com/science/article/pii/S0303264706002693

[17] E. Izhikevich, Polychronization: computation with spikes, Neural computation 18 (2) (2006) 245–282.

[18] S. Hartmann, S. Schiefer, S. Scholze, J. Partzsch, C. Mayr, S. Henker, R. Schiiffny, Highly integrated packet-based aer communication infrastructure with 3gevent/s throughput, in: Electronics, Circuits, and Systems

(ICECS), 2010 17th IEEE International Conference on, 2010, pp. 950–953. doi:10.1109/ICECS.2010.5724670.

[19] E. Culurciello, A. Andreou, A comparative study of access topologies for chip-level address-event communication channels, IEEE transactions on neural networks, vol. 14, no. 5, pp. 1266 - 1277.