AstroByte: Multi-FPGA Architecture for Accelerated Simulations of Spiking Astrocyte Neural Networks

Shvan Karim, Jim Harkin, Liam McDaid, Bryan Gardiner and Junxiu Liu
School of Computing, Engineering and Intelligent Systems
Ulster University, Magee Campus,
Derry, Northern Ireland, UK, BT48 7JL
{haji_karim-s, jg.harkin, lj.mcdaid, b.gardiner, j.liu1}@ulster.ac.uk

***Abstract*— Spiking astrocyte neural networks (SANN) are a new computational paradigm that exhibit enhanced self-adapting and reliability properties. The inclusion of astrocyte behaviour increases the computational load and critically the number of connections, where each astrocyte typically communicates with up to 9 neurons (and their associated synapses) with feedback pathways from each neuron to the astrocyte. Each astrocyte cell also communicates with its neighbouring cell resulting in a significant interconnect density. The substantial level of parallelisms in SANNs lends itself to acceleration in hardware, however, the challenge in accelerating simulations of SANNs firmly resides in scalable interconnect and the ability to inject and retrieve data from the hardware. This paper presents a novel multi-FPGA acceleration architecture, AstroByte, for the speedup of SANNs. AstroByte explores Networks-on-Chip (NoC) routing mechanisms to address the challenge of communicating both spike event (neuron data) and numeric (astrocyte data) across significant interconnect pathways between astrocytes and neurons. AstroByte also exploits the NoC interconnect to inject data and retrieve runtime data from the accelerated SANN simulations. Results show that AstroByte can simulate SANN applications with speedup factors of between x162 -x188 over Matlab equivalent simulations.**

Keywords*— *FPGA Acceleration, Multi-FPGA, Data acquisition, Networks on Chip, NoC, Astrocyte, SNN.

I. INTRODUCTION

The human brain can carry out computations in a power efficient and immensely parallel manner which has motivated the trend in bio-inspired computing [1]. Spiking Neural Networks (SNNs) are a popular bio-inspired paradigm that have been used in many applications [2]. The self-repairing ability of the human brain is a key attractive feature that engineers are keen to implement in the next generation of computers. In this context, current research in self-repair has focused on astrocytes, a type of glial cell, which is the mechanism responsible for facilitating fine-grained self-repair. These new Spiking Astrocyte-neuron Networks (SANNs) modulate the synaptic activities between neurons via distributed astrocytes in the network. This concept was proven in previous work when a biologically-faithful astrocyte was integrated with an SNN and simulated in Matlab [3]; this mechanism is illustrated in Fig. 1. When a synapse fails to release neurotransmitters, the associated neural activity falls and consequently the level of endocannabinoid (2-AG) decreases. The absence of the 2-AG signal, which is a retrograde feedback messenger from active postsynaptic neurons, causes an overall increase in probability of release (PR) at all tripartite synapses. This is because the direct feedback of 2-AG to the presynaptic terminal, which causes Depolarisation Induced Suppression of Excitation (DSE), has diminished leaving the indirect feedback signal from the astrocyte to cause a sudden increase in PR. This, in turn, increases the firing rate of the remaining synapses because a probabilistic tri-partite synapse has been used.

Subsequent research focussed on accelerating this entire process by means of designing a SANN accelerator on FPGA hardware [4]. A speedup factor of up to x1067 was achieved when compared with executing the model using Matlab. While designing the SANN accelerator, a configuration and monitoring platform was required, capable of passing configuration parameters to the hardware SANN and collecting monitoring data. This motivated the design of an FPGA Monitoring Platform (FMP) [5]. In this paper, the SANN accelerator and FMP are integrated into a multi-FPGA NoC platform called AstroByte.

The contributions of this work can be summarised as follows:

1- A new fully scalable NoC-based multi-FPGA architecture for accommodating SANN hardware acceleration.

2- Functional verification of the NoC infrastructure by integrating the SANN accelerator [4] on a 4-FPGA platform.

3- Integration of the FPGA Configuration and Monitoring Platform (FCMP) [5] with the multi-FPGA architecture, giving users the ability to configure the platform and acquire real-time simulation data.

4- Results demonstrating that SANNs can be accelerated using the NoC paradigm for facilitating cross-FPGA communications.

Simulating SNNs has traditionally been performed using a programming language such as MATLAB or PyNN that run on

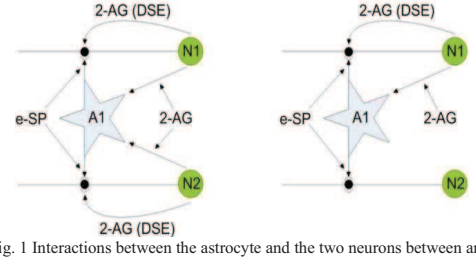


Fig. 1 Interactions between the astrocyte and the two neurons between and after a fault is injected [3].

single or multi-core general-purpose processors [6], [7]. In addition, some researchers have focussed on implementing SNN simulations on GPUs [8]. More recently, in-field reconfigurability and the ability to realise dedicated parallel hardware has made FPGAs attractive for SNN implementations [9]. To this effect, the EMBRACE architecture was developed, leveraging a 2D mesh NoC structure for scalable SNN FPGA simulations [10]. The EMBRACE router contains a programmable address table making simulation of different SNN structures possible. Bluehive [11] is a multi-FPGA platform for large-scale SNN real-time simulations.. The platform focusses on massive real-time simulations as opposed to accelerating these simulations. Each FPGA supports real-time simulation of 64,000 neurons and 64M synapses through time-multiplexing hardware resources. SNAVA [12], allows parallel real-time simulation of SNNs by means of a scalable and programmable architecture that can scale to incorporate a number of FPGAs. A shared bus extends to multiple FPGA boards in a ring topology using SMA cables and Xilinx Aurora protocol. SNAVA can be configured to a desired neuron and synapse model, utilising a time-multiplexing technique for taking full advantage of the available hardware.

The following distinguishes AstroByte from the above work:

1. Incorporation of astrocytes to provide SANNs with self-repairing abilities.
2. Ability to efficiently reconfigure the SANN and capture real-time simulation data using the same NoC infrastructure used for data communication.
3. Able to achieve accelerations of up to x188 while capturing full-scale real-time simulation data.

The rest of the paper is organised as follows. Section II investigates the overall architecture and operation of the AstroByte platform. Section III presents the results collected from different experimentations. At first the experiments setup will be explained followed by the results that include accuracy comparison with an equivalent software model, speedup gained and under-sampling. Section IV concludes the paper along with investigating future works.

II. AstroByte architecture

Fig. 2 shows the overall AstroByte architecture for a 4-FPGA system where four NoC routers are used to facilitate communication of configuration and simulation data. In effect, each Terasic DE4 FPGA board (Fig. 6) acts as a node on the NoC. Mesh topology is used along with a XY routing algorithm with a round-robin arbitration. XY routing has been utilized because of its efficiency –free of deadlock and livelock -and simplicity when used with mesh NoCs. In addition to its neighbouring routers, each router is also connected to an internal computing core. The computing core can be either an SANN building block (e.g. astrocytes, neurons, tripartite synapses, and spike generators) or the FCMP. In the case of the former the core is connected to a Core interface (CI) block and in case of the latter to the FCMP Interface (FI) block. The microarchitectures of the router and CI will be briefly discussed in the next subsections.

A. AstroByte protocol

Three types of packets are supported by AstroByte: data, credit, and configuration packets. Data and configuration packets are composed of two Flow Control Units (flit), a header flit and a body flit, while the credit packet is only one flit in size. Data packets are composed inside the cores and used for communicating neurons and astrocyte data with the other cores in the NoC. Configuration packets are sent from the FCMP to the other nodes on the NoC (other FPGAs) for configuring their operation. As shown in Fig. 3, data and configuration packets have similar structures with the difference being in the value of the four control bits in the header flit. When the control bits indicated a data packet, the header will be followed by a data flit that contains spike or astrocyte data. If the control bits had other values, a configuration packet is recognised, when the header flit is being followed by a flit that carries configuration values instead of spike of astrocyte values. Overall, AstroByte supports 12 configuration attributes. For example, in terms of reconfigurability, the user can choose the rate of injected faults, the time of occurrence of faults, and the destination of data packets from cores (i.e. mapping of the SANN). Credit packets are constantly sent to the neighbouring routers and their purpose is interchanging information on the available buffer space in the routers' input ports. This enable data to be only send when there is enough buffer space in the downstream router.

B. Router microarchitecture

Fig. 4 shows the router micro-architecture. Each router consists of five ports, four for communicating with neighbouring nodes/FPGAs and one internal port for the computing core. All four external ports are connected to a bidirectional Intel Gigabit Transceiver Block (GXB) for serialising the data stream and sending over SATA connections. GXB transceivers accept and supply parallel data in their clock domains generated by the receiver and transmitter of the GXB respectively. Although the frequency of these clocks is 150MHz (i.e. similar to the router frequency), their phase is different, thus, should be dealt with as different clock domains. Because of existence of many frequency domains in the router, the Clock Domain Crossing (CDC)

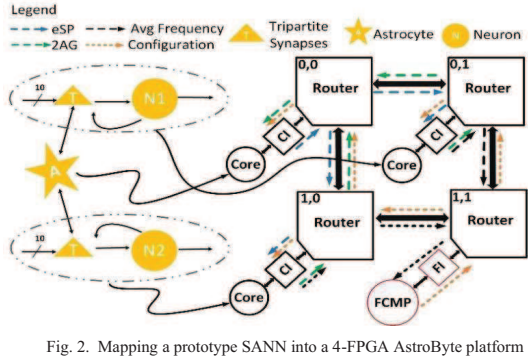


Fig. 2. Mapping a prototype SANN into a 4-FPGA AstroByte platform

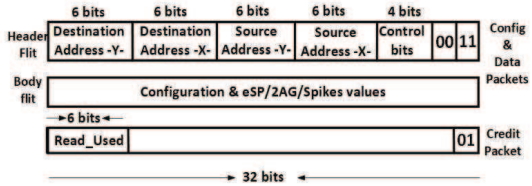


Fig. 3. AstroByte protocol format

technique has been utilised using Dual Clock First in First out (DCFIFO) memories and associated controllers. The micro-architecture is illustrated in Fig. 4, where ports 1-4 have identical building blocks while port 0, the core port, has a different structure. The core port is different because its input and output buses are connected to the CI which is located on the same FPGA, meaning that GXBs are not required for serialising Data. The input controller of Fig. 4 is responsible for de-multiplexing the input stream and forwarding data to the two DCFIFOs; one is data packet buffer (Data DCFIFO) and the other is credit packet buffer (Credit DCFIFO). The Input controller and the write side of the DCFIFOs operate in the GBX Rx clock domain which is different for each port. Control logic 1 (CL1), the read side of the input DCFIFO, and CL2, the write side of output DCFIFO, operate in the router clock domain. CL1 contains controller logic handling reading from the Data DCFIFO and generating request signals by decoding the address in the header packets existing at the DCFIFO outputs. Furthermore, CL1 contains logic for deciding whether the Arbiter can grant request on the output port. CL1 decided this firstly, decoding credit packets from the Credit DCFIFO and determining if the downstream router can accommodate further packets. For the internal port, no credit packet is received and decoded. Instead, as the CI is located on the FPGA, information regarding free buffer space is obtained by a separate bus, denoted as *CI Write_Used* in Fig. 4. Secondly CL1 uses the *Write_Used* signal from output Data DCFIFO to decide if the output buffer has free space available. This information is passed to the Arbiters (one in each port) that control access to the output ports in the crossbar switch. The Arbiter implements a round-robin arbitration scheme and it will only grant access to requesting ports if the CL1 has determined there is free space in both the output buffer of the current router and the input buffer of the next router (or the internal node). CL2 contains logic to packetize the *Read_Used* signal (indicating the number of used words) from the input Data DCFIFO to form the credit packet. Also, CL2 is responsible for controlling write operations to the two output DCFIFOs, one for data packets and credit packets each. Output controller along with the read side of output DCFIFOs operate in the GXB Tx clock domain. Output controller multiplexes both data packet and credit packets before forwarding the data stream to the transmitter parallel input.

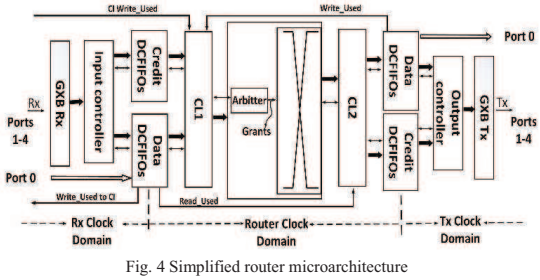


Fig. 4 Simplified router microarchitecture

C. Core Interface (CI) microarchitecture

The CI block is located between the internal port of the router-port 0 and the computing core. In Fig. 5, the left side of CI communicates with the router while the right side communicates with the core which is the SANN computation. The Input Controller block decodes the header in the incoming packet from the router and forwards the body part to either Input Data DCFIFOs or configuration registers. If a data packet is detected, the header will be discarded of and the body, which is the data flit, will be stored in the Input Data DCFIFOs. If a configuration packet is recognized, the header will be decoded by the Input controller and the body forwarded to the relevant register in either the Configuration Register File (CRF) or CRF DCFIFO. The former parameters are used by the Output controller and the latter will be passed to the SANN core. The Output controller packetizes the core data, saved in the Output Data DCFIFOs and forwards it to the router, i.e. as long as port 0 has free buffer space. The output controller sends data according to the content of CRF. For example, while packetizing data, the Output controller uses the value stored in the CRF address register, received at the start of operation, as the destination address of the packets. Additionally, the value stored in the under-sampling register of CRF will be used to decide whether to send data at full or reduced granularity.

D. FCMP & FI

FCMP is a Nios II embedded processor-based system that can communicate to a PC and the FPGA fabric. FCMP is placed on one of the FPGAs with a NoC router. This allows the FCMP to address and be addressed by the other FPGAs, allowing for easy configuration and monitoring as the FCMP will act similar to the other nodes on the network. The FCMP & FI microarchitecture is an adaptation of the design reported in [5].

E. AstroByte Operation

Initially all FPGA boards are programmed with SANN cores and a single FCMP HDL codes. The configuration phase starts next, in which the FCMP node starts sending configuration

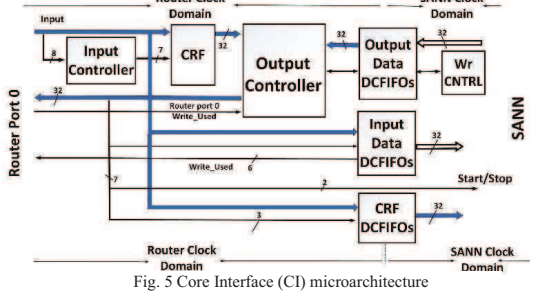


Fig. 5 Core Interface (CI) microarchitecture

packets to the other nodes to facilitate a user defined multi-FPGA SANN platform through modifying the SANN parameters. As an example, if inserting faults was required by the simulation, appropriate configuration packets can be sent to specify the rate of the fault and the time of occurrence in clock cycles. Next, the FCMP sends a special configuration packet that carries start command for starting acceleration at the SANN cores. The SANN cores then commence sending data through the NoC infrastructure to destinations specified by the configuration packets. The destinations are other nodes with the possibility of sending data to FCMP for monitoring. When the FCMP SRAM is full, a trigger goes to the Nios II processor and sending data from the SRAM to a PC begins. The data will be sent using Ethernet to a PC that runs Matlab for storage, monitoring and analysis purposes.

III. Experimentations and Results

A. Experiment setup

Intel Quartus Prime 18.0 SE was used for FPGA design, synthesis and programming. SignalTap II and the in-house developed FCMP were used for design verification. Matlab R2015b was used for capturing simulation data for storage and analysis. An Intel build for Eclipse Mars 2 was used with Nios ii embedded processor. All software simulation experiments were executed on a 3.4 GHz Intel Core i7-2600 with 16GB of RAM, running 64-bit Windows 10.

Terasic DE4 FPGA boards , each with an Intel Stratix IV GX EP4SGX530 FPGAs, were used to implement AstroByte, as can be seen in Fig. 6. DE4 boards provide a number of standard interfaces and AstroByte uses Ethernet, SATA, and General Input Output Pin (GPIO) interfaces.

The SATA links provide a throughput of 6Gbps. Although the transceivers can support this throughout, the maximum effective throughput across the link is 4.8Gbps due to a 8b/10b encoding scheme [13]. In this work, 2.7Gbps (70%) of the bandwidth has been utilised.

B. Multi-FPGA SANN Implementation

The set of experiments that will presented next aim at evaluating performance of the AstroByte platform by integrating the SANN accelerator block into the multi-FPGA NoC architecture. Fig. 2 illustrates this integration. The first neuron entity (N1), and the associated tripartite synapses, spike and probability generators, are mapped into node (0,1) in the NoC mesh. N2 and its associated blocks are mapped to node (1,0) while the astrocyte core is mapped to node (0,0) and FCMP is located at node (1,1). FCMP sends configuration packets at the start of a simulation, routing information from the SANN components to their destination. The Astrocyte core sends eSP signals to both neuron cores, which in turn, send the astrocyte 2AG values. The neurons also send their average frequencies to the FCMP platform, enabling users to monitor the rate of firing.

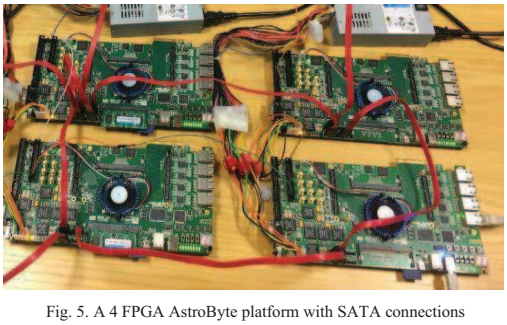


Fig. 5. A 4 FPGA AstroByte platform with SATA connections

C. Acceleration

Acceleration gained from the prototype AstroByte platform of Fig. 2 will be assessed in this section. Table. 1 shows that SANN speedup factors of between x162 and x188 can be gained on with AstroByte compared to an equivalent Matlab model [6]. Comparing these figures with acceleration obtained from a single FPGA SANN platform indicates that, despite the NoC interconnection overhead, the multi-FPGA AstroByte can maintain over 75% of the speedup factor. From the last entry in Table.1, 2hrs:46min of biological time can be simulated in 21.74 seconds using the proposed multi-FPGA accelerator. The same biological time scale will take 1hr: 8mins in Matlab.

D. Accuracy

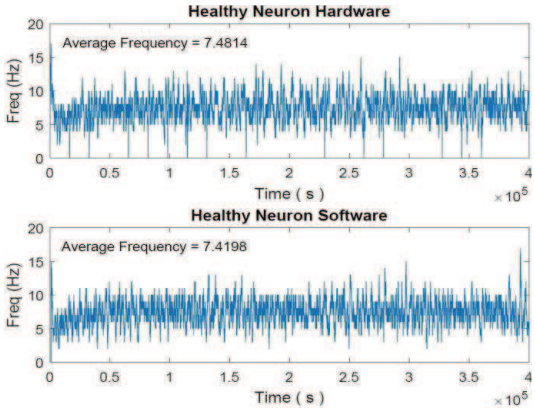
Fig. 7 illustrates average frequency comparison between AstroByte SANN implementation and equivalent Matlab software model. Frequency response trajectories for the healthy neurons and various rations of faults, including no fault situation, are presented. The average frequency of the trajectories is also shown. It is evidence that a multi-FPGA AstroByte platform can simulate SANNs with high accuracy. The maximum difference between the average two frequencies can be seen in 7(f), with a difference of 0.0939 Hz (~0.016%).

E. Under-sampling

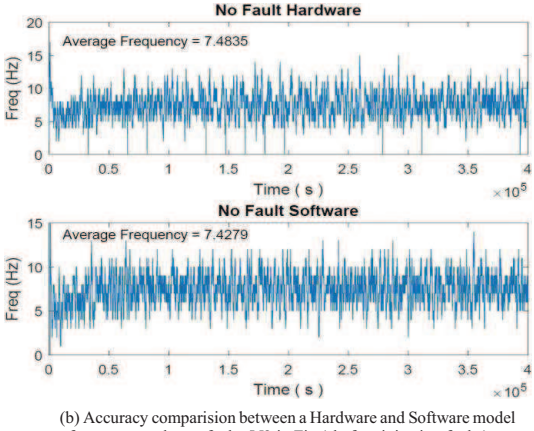
Under-sampling is carried out by collecting the same amount of data with reduced granularity, resulting in faster simulation and data acquisition. Table. 2 demonstrates increased speedup from under-sampling.

TABLE.1.Comparison between different SANN implementations

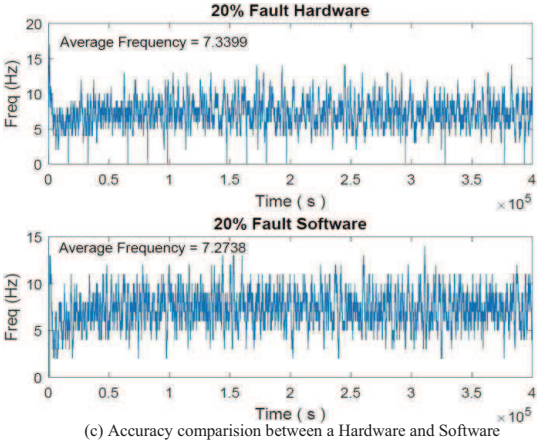
Biological Time(s)	*Iterations (Cycles)*	*Matlab (Seconds)*	*AstroByte (Seconds)*	*Acceleration*
400	400,000	153.72	~0.938	163.88
600	600,000	242	~1.4	172.85
800	800,000	327	~1.9196	170.34
1,000	1,000,000	381	~2.3448	162.48
1,200	1,200,000	506	~2.71	186.71
3,600	3,600,000	1500	~8.1618	183.78
5,000	5,000,000	1960	~10.7723	181.96
10,000	10,000,000	4095	~21.7448	188.32



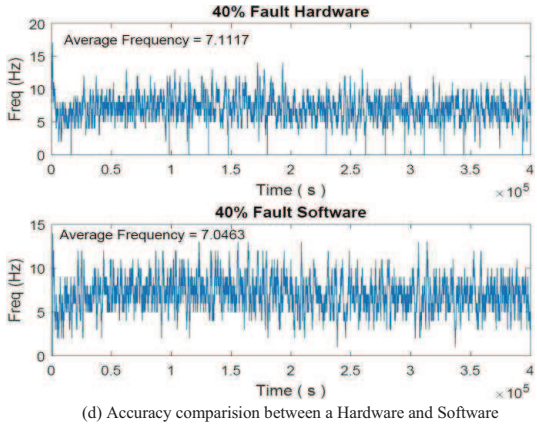
(a) Accuracy comparision between a Hardware and Software model of a healthy neuron (N1 in Fig.1)



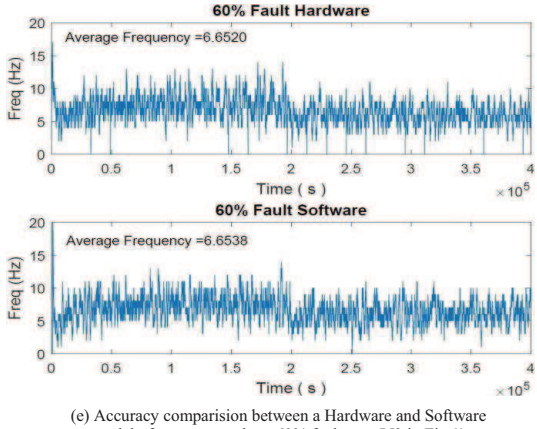
(b) Accuracy comparision between a Hardware and Software model of a neuron under no fault (N2 in Fig.1 before injecting faults)



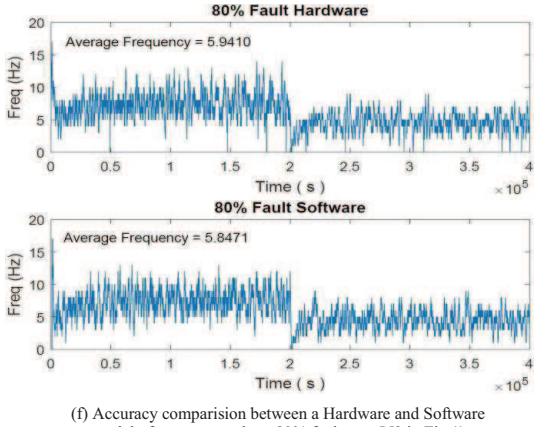
(c) Accuracy comparision between a Hardware and Software model of a neuron under a 20% fault rate (N2 in Fig.1)



(d) Accuracy comparision between a Hardware and Software model of a neuron under a 20% fault rate (N2 in Fig.1)



(e) Accuracy comparision between a Hardware and Software model of a neuron under a 60% fault rate (N2 in Fig.1)



(f) Accuracy comparision between a Hardware and Software model of a neuron under a 80% fault rate (N2 in Fig.1)

Fig. 6 Accuracy comparison between AstroByte (Hardware) and Matlab (Software) for different fault rates

TABLE.2 UNDER-SAMPLING RESULTS

Granularity	*Elapsed time (Seconds)*	*Iteration/Node (Cycles)*	*Biological time scale (Seconds)*
1	1044.8	401,408,000	401,408
10	158.14	401,408,000	401,408
100	88.06	401,408,000	401,408
1,000	82.76	401,408,000	401,408

Using a granularity of 1, i.e. all processed data is sent to the FCMP for monitoring, simulating 401,408,000 cycles per node can be performed in 17 minutes and 24 seconds. The same 'biological-scale' of simulation can be carried out in approximately 2 minutes and 38 seconds when the granularity of the data captured is 10 (one in every ten data-points of simulation is collected). Reducing the granularity to 100 will result in simulating and capturing monitoring data in approximately 1 minute and 28.06 seconds. The advantage of under-sampling in is increased speed-up as the higher granularity means less data is transferred to the monitoring PC, resulting in the SANN working closer to its maximum acceleration capability. Given that the astrocyte is a slow biological process with dynamic changes very slow in nature (in the order of tens of seconds), the impact on accuracy of simulation is not significant [5] when samples are dropped.

IV. CONCLUSION AND FUTURE WORKS

This paper introduced AstroByte, a novel multi-FPGA, scalable and programmable platform for accelerating Spiking Astrocyte Neural Network applications. The overall AstroByte architecture was detailed along with the Networks-on-Chip mechanisms used to support the dense interconnect (exchange of data) between astrocytes and neurons. Results demonstrated that the platform can accelerate SANN applications with speedups of up to x188 over an equivalent Matlab model. It was shown that the platform can replicate simulation traces (behaviour) obtained from Matlab with high accuracy (a maximum difference of 0.016%). The under-sampling feature was examined, and its advantages and limitations were discussed in terms of increasing speedup with minimal loss of accuracy.

Future work includes exploring the implementation of large-scale SANN models (10^2 astrocyte cells) incorporating significant neurons. Additionally, further performance optimisations will be investigated in terms of acceleration through astrocyte process optimisation and further architecture exploration.

ACKNOWLEDGMENT

The authors would like to acknowledge the EPSRC funding council grants (EP/N00714X/1 & EP/N007050/1) and Ulster University for supporting this research.

REFERENCE

[1] Q. Chen and Q. Qiu, "Real-time anomaly detection for streaming data using burst code on a neurosynaptic processor," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, 2017, pp. 205–207.

[2] A. Das, Y. Wu, K. Huynh, F. Dell'Anna, F. Catthoor, and S. Schaafsma, "Mapping of local and global synapses on spiking neuromorphic hardware," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2018, pp. 1217–1222.

[3] J. Wade, L. McDaid, J. Harkin, V. Crunelli, and S. Kelso, "Self-repair in a bidirectionally coupled astrocyte-neuron (AN) system based on retrograde signaling.," *Front. Comput. Neurosci.*, vol. 6, no. September, p. 76, 2012.

[4] S. Karim *et al.*, "Assessing Self-Repair on FPGAs with Biologically Realistic Astrocyte-Neuron Networks," in *Proceedings of IEEE Computer Society Annual Symposium on VLSI, ISVLSI*, 2017, vol. 2017-July, pp. 421–426.

[5] S. Karim *et al.*, "FPGA-based Fault-injection and Data Acquisition of Self-repairing Spiking Neural Network Hardware," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018, pp. 1–5.

[6] A. P. Davison, "PyNN: a common interface for neuronal network simulators," *Front. Neuroinform.*, vol. 2, 2008.

[7] I. Bogdanov, R. Mirsu, and V. Tiponut, "MATLAB model for spiking neural networks," *Proc. 13th WSEAS Int. Conf. Syst.*, no. July 2009, pp. 533–537, 2009.

[8] A. K. Fidjeland and M. P. Shanahan, "Accelerated simulation of spiking neural networks using GPUs," *Int. Jt. Conf. Neural Networks*, pp. 1–8, 2010.

[9] D. Ferrer, R. Gonzalez, R. Fleitas, J. P. Acle, and R. Canetti, "NeuroFPGA-implementing artificial neural networks on programmable logic devices," in *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, pp. 218–223.

[10] L. McDaid, J. Harkin, S. Hall, and T. Dowrick, "EMBRACE: emulating biologically-inspired architectures on hardware," *NN'08 Proc. ...*, 2008.

[11] S. W. Moore, P. J. Fox, S. J. T. Marsh, A. T. Markettos, and A. Mujumdar, "Bluehive - A field-programable custom computing machine for extreme-scale real-time neural network simulation," in *Proceedings of the 2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines, FCCM 2012*, 2012, pp. 133–140.

[12] A. Sripad *et al.*, "SNAVA—A real-time multi-FPGA multi-model spiking neural network simulation architecture," *Neural Networks*, vol. 97, pp. 28–45, 2018.

[13] A. X. Widmer and P. A. Franaszek, "A DC-Balanced, Partitioned-Block, 8B/10B Transmission Code," *IBM J. Res. Dev.*, vol. 27, no. 5, pp. 440–451, Sep. 1983.