FPGA-based Fault-injection and Data Acquisition of Self-repairing Spiking Neural Network Hardware

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

David M. Halliday, Andy M. Tyrrell, Jon Timmis, Alan G.
Millard and Anju P. Johnson
Department of Electronic Engineering,
University of York, Heslington,
York, UK, YO10 5DD
{david.halliday, andy.tyrrell, jon.timmis, alan.millard,
anju.johnson}@york.ac.uk

Abstract—**Spiking Astrocyte-neuron Networks (SANNs) model the adaptive/repair feature of the human brain. They integrate astrocyte cells with spiking neurons to facilitate a distributed and fine-grained self-repair capability at the synapse level. SANNs are more complex with the addition of astrocyte cells and require longer simulation times, as they are dynamic over much longer time-scales than traditional neural networks. Therefore, dedicated FPGA accelerators offer reductions in simulation times. To support the acceleration of SANNs, the capability of fault injection to synapses and monitoring significant levels of neuron and astrocyte data for off-chip transmission to PC-based analysis, are required. This paper presents an FPGA-based monitoring platform (FMP) for injecting faults and capturing and analyzing data acquired from the SANN FPGA accelerator, Astrobyte. The FMP uses custom logic and a NIOS II based system to control fault injection and data monitoring on the FPGA. Results show accurate accelerated simulations of fault injection scenarios using FMP with speedups up to 65 times greater compared with equivalent Matlab implementations.**

Keywords—FPGA acceleration; Data Acquisition; Astrocytes; Spiking neural network ; Self repair; Fault injection

I. Introduction

The human brain can carry out computations in a power-efficient and massively 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 an astrocyte was integrated with an SNN [3]. Due to the complex nature of the astrocyte model in previous work [3], simulating the SANN using tools such as Matlab required long times, in particular as the astrocyte is not event based like traditional SNNs and operates over longer biological timescales of hundreds of seconds. This motivated the provision of Astrobyte, an FPGA-based platform for accelerating simulations of SANNs through implementing dedicated astrocyte, neuron and synapse hardware models on FPGAs [4]. To facilitate experimentation, a framework is required to enable fault injection, spike/astrocyte data recording and visualization of FPGA-based SANNs. Current mechanisms such as Altera SignalTap II Logic Analyzer is not adequate as its capacity is limited by the amount of memory provided by the FPGA on-chip memory.

There are multiple FPGA-based acquisition platforms in the literature. One work has used an FPGA as a bridge between an Analogue to Digital Converter (ADC) and an off-chip DDR3 SDRAM [5]. However, it is not clearly stated how the data stored in the DDR3 SDRAM will be analyzed as no method of transferring this data to a PC is mentioned. A different publication uses a Xilinx Spartan 6 FPGA for acquiring data from an ADC and sending it to a PC by using Ethernet for monitoring [6]. This is similar in concept to the work in the current paper but has been implemented using Xilinx FPGAs and tools. Also, the application proposed in [6] focusses on dedicated imagers in nuclear medicine as opposed to SANNs, the focus of the current paper. Another FPGA-based monitoring platform has included an FPGA for the purpose of signal processing and controlling multiple sensor channels in a machine condition monitoring system [7]. Here the FPGA doesn't include any form of soft processor, but instead controls a number of monitoring channels while processing data at the same time. The processed data is then sent to a PowerPC based control system which in turn sends the data to a monitor.

Improving on the work reported in [4], the novelty of the work proposed in this paper resides in the provision of a framework that is able to inject faults into SANNs on FPGA hardware and acquire real-time network data from Astrobyte [4] by means of the FMP. The rest of the paper is organized as follows: Section II describes the architecture and operation of the FMP. In Section III experiments and results are presented and Section IV provides a conclusion along with discussing future works.

II. Architecture and Operation

A. Components

Fig. 1 shows the overall architecture of the platform reported in this paper. Except for the Astrobyte block, all other components are part of the FMP (FMP itself contains two main blocks, Astrobyte Interface and Nios II system). Besides the main blocks, the FMP also contains several off-chip components

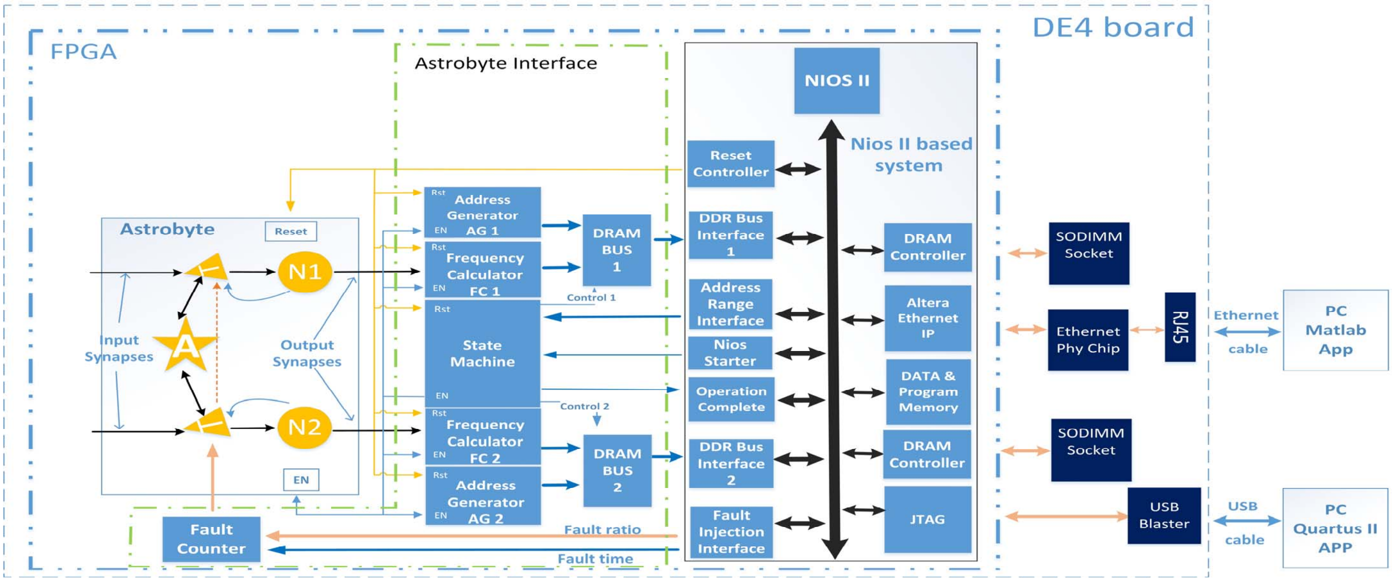

Fig 1: FMP Architecture and Astrobyte

that are located outside the FPGA. A brief description of Astrobyte and the FMP blocks are given below.

The Astrobyte [4] is a custom FPGA design that accelerates the simulation of a self-repairing SANN based on neuron, astrocyte and synapse models reported in [3]. In these works [3], [4], an astrocyte is included in an SNN for regulating the synaptic activity of neurons at tripartite synapses. Each neuron is fed from ten synapses. Faults are injected at a specific time to a number of the synapses and the average output frequencies of the two neurons, N1 and N2 in Fig. 1, are monitored. It was verified that the network can recover from faults due to the astrocyte since it compensates for lack of activity at the synapses that suffered the fault.

The Astrobyte Interface contains several hardware sub-blocks. The two Frequency Calculators (FC) calculate the average spiking frequencies of the neurons over a time interval. The two Address Generators (AG) are essentially counters with an enable signal and a parameterizable start value. This is necessary because the Nios II based system operates as a memory mapped system thus, at least one of the AGs has to start from a base value other than zero. The State Machine is responsible for controlling the operation of Astrobyte and Astrobyte Interface. It exchanges control and status signals with the Nios II based system. The state machine operation will be discussed in more detail in Section II.C. The DRAM BUS blocks are combinational blocks that combine addresses generated from AGs and data from FCs along with control signals from the state machine to form a complete bus that writes into the DRAM controllers.

The Nios II based system is a heavily modified version of the simple socket server design example provided by Terasic for use with DE4 boards. The Nios II based system has an Altera Nios II soft-core processor at its heart. The system also includes a Data and Program Memory which is an SRAM, an Altera Ethernet IP which is responsible for sending Ethernet packets to an off-chip PHY chip, and DRAM Controllers for communicating with off-chip DDR2 SDRAMs. Additionally, a JTAG IP that allows the system to be programmed from a PC using Eclipse environment, and a number of command and interfacing blocks are also included in the Nios II system. The command and interfacing blocks allow the Nios II processor to communicate with the Astrobyte Interface components for passing data, status, address and control signals. The command and interfacing blocks are Reset Controller, Nios Starter, Address range Interface, Fault Injection Interface, DDR Bus Interface and Operation Complete. The purpose of these blocks will become clear in Section II-C.

The architecture of Fig.1 allows for control over the start and end time of Astrobyte simulations besides fault injections and capturing simulation data - average output frequencies - and sending it to a PC.

B. *Datapath*

As per Fig. 1, the two outputs from the Astrobyte accelerator platform represent the synapses of the two neurons – N1 and N2. These outputs feed into the two FC blocks. Outputs from FCs will be combined with addresses generated by AGs and control signals from the state machine at the DRAM BUS blocks. Next, the DRAM BUS output has to go through the DDR BUS Interface block since an interface is required to transfer data to the Nios II system. Also, the outputs of DDR BUS Interfaces will become part of a unified Avalon (Altera standard interface) bus before entering the DRAM Controller. The DRAM Controller takes address, data and control signals and generates appropriate outputs so that data is transferred to and from the external DDR2 SDRAM.

C. *Operation*

A flow chart is given in Fig. 2 explaining the operation of the FMP. The Nios II processor controls the operation of the custom hardware circuit by means of a set of parameters, control and reset signals which are passed during the reset state. One of these parameters is the fault ratio, which is the number of damaged synapses to the total number of synapses in Astrobyte. Another parameter is the time at which the faults acquire (through the

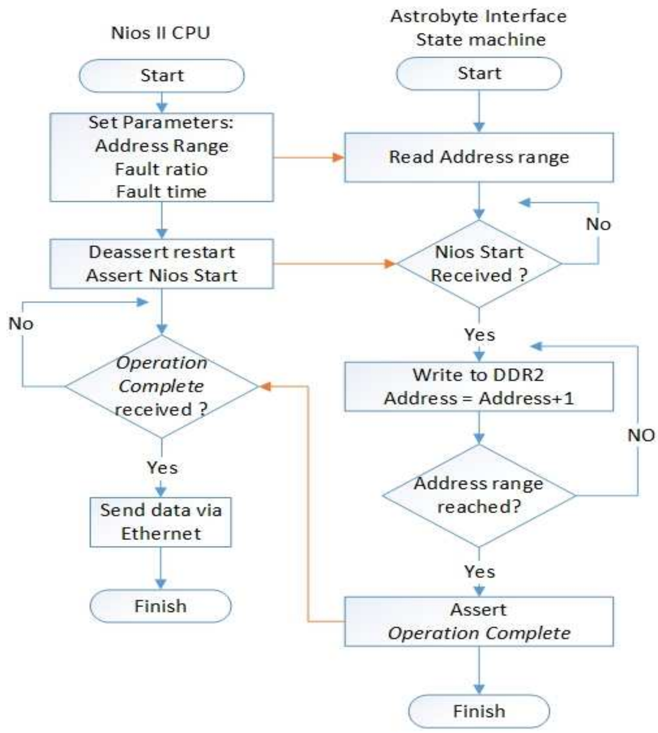

Fig 2: Algorithm outlining the FMP operation

Fault Injection Interface in Fig. 1). Moreover, the amount of data to be written into the DDR2 SDRAMs can also be passed to the Astrobyte Interface through the Address Range Interface. This value equals the number of cycles Astrobyte runs for. Furthermore, through the Nios starter block in Fig. 1, the Nios II processor sends a start signal to the state machine. Once this command is received, the state machine enables the FCs and AGs along with issuing write commands to the DRAM controllers. This commences writing of the average frequency data from the FCs into the external DDR2 SDRAMs at addresses generated by the AGs. This process continues until data is written into a range of addresses defined by the parameter passed from the Address Range Interface at the reset state. After this process, the state machine sends Nios II a signal through the Operation Complete block and the Nios II processor starts to send data from the external DDR2 SDRAMs to a PC through Ethernet. Fig. 3 shows a simplified sequence diagram for passing of commands and data between Matlab, Nios II and the Astrobyte hardware.

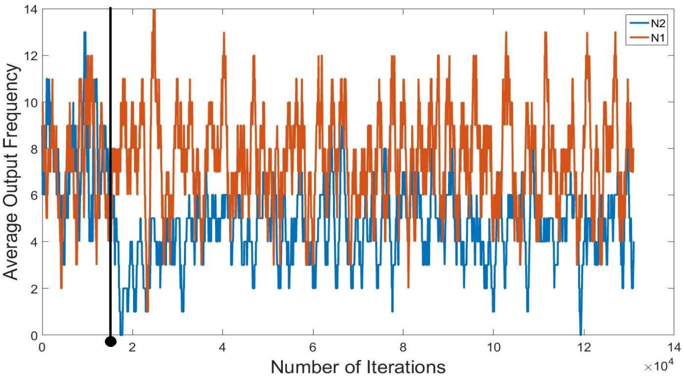

Fig. 4: Data collected using the FMP and SignalTap II

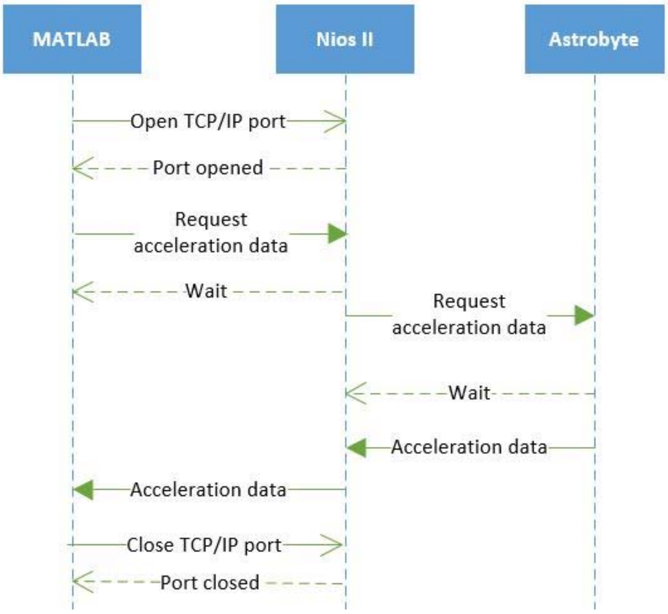

Fig 3: Interactions between Matlab, Nios II and Astrobyte

III. EXPERIMENTATIONS AND RESULTS

A. Assessing data integrity

Several experiments were performed to ensure that data captured by using the FMP is accurate and remains undistorted while transferred from the Astrobyte to external memory, and then onto the PC via Ethernet. Matlab software was used to communicate with the Nios II soft processor which managed the fault injection, recording the data communication between the SANN and SDRAM. To validate the data management, the same experiment that was carried out in [4] and recorded using SignalTap II was repeated here using the FMP. Fig 4 show plotted data collected from Altera SignalTap II and the FMP. The *x-axis* is the number of clock cycles, or iterations, and the *y-axis* represents the average output frequency of Astrobyte. The simulations were initially run without injecting faults. Subsequently, faults were inserted, damaging 80% of synapses connected to N2. As the FMP had not been developed in previous work, Altera In-System Sources and Probes Editor had to be used to inject faults. The timing of these faults had to be decided before synthesizing the design from the HDL code. In the current work, the FMP allows for inserting faults at different ratios at a time chosen by the user. The faults are indicated by a black vertical line in Fig. 4 and Fig. 5. At first, neuron N2 will see a sharp decrease in average output frequency but then recovers because of the self-repair mechanism that is regulated by the astrocyte. Since the fault ratio is very high, i.e. 80%, the average output frequency does not go back to its pre-fault levels. Data from this experiment was recorded after it was transferred to the PC side. Then it was plotted against data recorded by using SignalTap II in the original Astrobyte paper [4]. Fig. 4 shows data acquired using the FMP. As no loss of data happens when using the FMP, acquiring data using Altera SignalTap II yields a similar response. Fig. 5 shows results from the Matlab implementation of SANN. Clearly, Fig. 4 and Fig. 5 show the same astrocyte behavior, with the marginal difference in the trajectories of data presented in the figures due

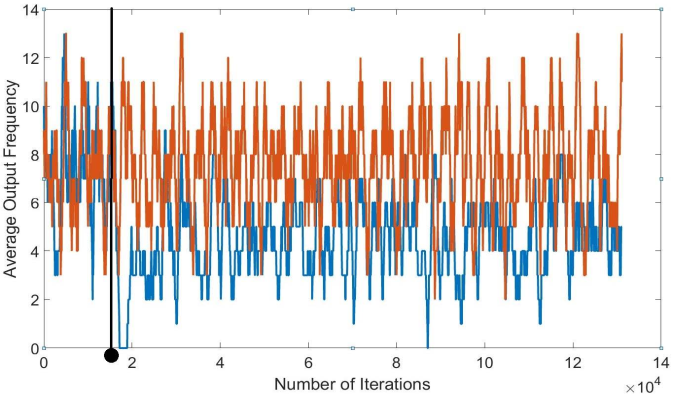

Fig. 5: Matlab implementations results

to the difference in implementations, i.e. the Matlab model uses double point floating-point precision and Astrobyte uses an area optimized 32-bit fixed-point hardware implementation [4].

B. Acceleration

Table. 1 presents a comparison between simulating a SANN using Matlab (software) and Astrobyte platform (dedicated FPGA hardware). The Biology column represents the actual biological time-scale of the simulations. Iterations is the number of times the equations representing the SANN must be calculated or the number of cycles Astrobyte needs to run to meet the set biological time-scale. This value is x1000 the biological time since a time step of 10^{-3} is selected [3] for the Euler method. Both Matlab and Astrobyte columns show how much time a Matlab software model [3] and an equivalent FPGA accelerator [4], respectively, take to run the simulation for the corresponding biological time-scale. The Matlab software runs on Windows 10 on a PC with 256 GB SSD, 16 GB RAM and 3.40 GHz Intel Core i7-2600 CPU (Octa-Core). Also, the table shows the time it takes the FMP to transfer data generated by Astrobyte to a PC, shown under the FMP column. The column Astrobyte + FMP provides the total time from the start of running the design in Astrobyte to collecting the data on the PC side. The Speedup column shows the speedup gained by using Astrobyte and the FMP, in comparison to the Matlab software. The overall speedup is in the order of x50-65 depending on the number of iterations the designs are run for. It is worth mentioning that the values under Matlab, FMP, Astrobyte + FMP and Speedup can vary from one PC to another and also from time to time as they are PC and Windows OS dependent. Studying Table. 1 shows that using the FMP to transfer simulation data from the FPGA to a PC will introduce a significant overhead to the Astrobyte time. However, up to x65 speedup rate is possible which is significant for a wide range of applications including the study of how astrocytes impact on other neurological conditions such as Alzheimer's [8].

C. Reducing sampling rate

When simulating biology, it is possible to drop the sampling rate to one sample per ten computations or one sample per hundred computations as the rate of change in biology is slow. For example, astrocyte dynamics operate in hundreds of seconds. That would allow simulating the SANN for longer periods while maintaining significant speedup since we would need to transfer less data to the PC side. Table. 2 shows the effect of under-sampling for biology time of 100,000 seconds. For example, if every one in ten samples are recorded, the overall simulation and acquisition time will be reduced from around 724 seconds to ~63 seconds. This provides a significant reduction in the time communicating data off-chip (FPGA) back to the PC. This feature allows the Astrobyte platform to vary the accuracy of simulations as required. In exploring the self-repair aspect for fault tolerant networks can require less samples such as discussed above, however, for simulating a neural experiment to study astrocyte dynamics between neurons for example, can require the higher data sampling rate. The under-sampling provides a trade-off between simulation accuracy and speedup capability.

TABLE 1. Comparison between different implementations

Biology (Sec)	*Iterations (Cycles)*	*Matlab (Sec)*	*Astro bye (Sec)*	*FMP (Sec)*	*Astrobyte + FMP (Sec)*	*Speedup*
400	400 K	153.72	0.04	~3	~3.04	50.5
1,000	1 M	381.5	0.1	~6.7	~6.8	56.0
5,000	5 M	1960	0.5	~30	~30	65.2
10,000	10 M	4095	1	~62	~63	65.0

IV. Conclusions and Future works

In this paper, an FPGA – based Monitoring Platform (FMP) was presented which captures data from an FPGA accelerator platform, Astrobyte. Comparisons were made between SANNs implemented in Astrobyte FPGA and Matlab software, to assess the overall hardware speedup. Results demonstrated over x65 speedup using Astrobyte with the FMP. Finally, using FMP, an analysis from the effects of under-sampling in Astrobyte were discussed which showed possible trade-offs between reduced simulation accuracy and increased speedup. Future work will include exploring methods for scaling the FMP and Astrobyte to partition across multiple FPGAs, investigating methods for speeding up acquisition and transformation of data, and adapting the FMP for other neural network models other than SANNs.

Acknowledgements

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

TABLE. 2 Under-sampling evaluation

Sampling	*Iterations (Cycles)*	*Astrobyte (Sec)*	*FMP (Sec)*	*Astrobyte + FMP (Sec)*
1/1	100 M	10	~714	~724
1/10	10 M	1	~62	~63
1/100	1 M	0.1	~6.7	~6.8

REFERENCES

[1] Q. Wu, B. Liu, Y. Chen, H. Li, Q. Chen, and Q. Qiu, “Bio-inspired computing with resistive memories - Models, architectures and applications,” Proc. - IEEE Int. Symp. Circuits Syst., pp. 834–837, 2014.

[2] S. R. Kulkarni, A. V Babu, and B. Rajendran, “Spiking Neural Networks – Algorithms , Hardware Implementations and Applications,” no. 1, pp. 426–431, 2017.

[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,” Proc. IEEE Comput. Soc. Annu. Symp. VLSI, ISVLSI, vol. 2017–July, pp. 421–426, 2017.

[5] A. A. Khedkar and R. H. Khade, “High speed FPGA-based data acquisition system,” Microprocess. Microsyst., vol. 49, pp. 87–94, 2017.

[6] E. Fysikopoulos, G. Loudos, M. Georgiou, S. David, and G. Matsopoulos, “A Spartan 6 FPGA-based data acquisition system for dedicated imagers in nuclear medicine,” Meas. Sci. Technol., vol. 23, no. 12, p. 125403, 2012.

[7] I. Humphreys, G. Eisenblätter, and G. E. O’Donnell, “FPGA based monitoring platform for condition monitoring in cylindrical grinding,” Procedia CIRP, vol. 14, pp. 448–453, 2014.

[8] J. J. Wade, L. J. McDaid, J. Harkin, V. Crunelli, and J. A. S. Kelso, “Bidirectional coupling between astrocytes and neurons mediates learning and dynamic coordination in the brain: A multiple modeling approach,” PLoS One, vol. 6, no. 12, pp. 1–24, 2011.