Fault-tolerant Learning in Spiking Astrocyte-Neural Networks on FPGAs

Anju P. Johnson†, Junxiu Liu*, Alan G. Millard†, Shvan Karim*, Andy M. Tyrrell†,
Jim Harkin*, Jon Timmis†, Liam McDaid* and David M. Halliday†
†Department of Electronic Engineering, University of York, York YO10 5DD, UK
∗School of Computing and Intelligent Systems, Ulster University, Derry BT48 7JL, UK
Email: {anju.johnson, alan.millard, andy.tyrrell, jon.timmis, david.halliday}@york.ac.uk
Email: {j.liu1@, haji_karim-s@email., jg.harkin@, lj.mcdaid@}ulster.ac.uk

***Abstract*—The paper presents a neuromorphic system implemented on a Field Programmable Gate Array (FPGA) device establishing fault tolerance using a learning method, which is a combination of the Spike-Timing-Dependent Plasticity (STDP) and Bienenstock, Cooper, and Munro (BCM) learning rules. The rule modulates the synaptic plasticity level by shifting the plasticity window, associated with STDP, up/down the vertical axis as a function of postsynaptic neural activity. Specifically when neurons are inactive, either early on in the normal learning phase or when a fault occurs, the window is shifted up the vertical axis (open), leading to an increase in firing rate of the postsynaptic neuron. As learning progresses, the plasticity window moves down the vertical axis until the desired postsynaptic neuron firing rate is established. Experimental results are presented to show the effectiveness of proposed approach in establishing fault tolerance. The system can maintain the network performance with at least one nonfaulty synapse. Finally, we discuss a robotic application utilizing the proposed architecture.**

***Keywords*—Neuromorphic Computing, Fault Tolerance, Self-Repair, Spiking Neural Network, Astrocyte, Field Programmable Gate Array, Bio-inspired Engineering.**

I. Introduction

Neuromorphic computation is an emerging research domain, which derives inspiration from the architecture of nervous systems of living beings to solve complex tasks. Spiking neurons are core components of many computational models of the brain that aim to improve understanding of brain function. Although neuromorphic systems based on Very Large Scale Integration (VLSI) architecture were introduced in the late 1980s [1], use of FPGAs in the design of spiking neuromorphic architectures started to be used over the last ten years [2]. In Spiking Neural Networks (SNNs), communication and computation happen by an exchange of spatiotemporal patterns encoded as spikes as in biological neurons. Astrocytes have been shown to coexist with neurons [3] where these cells communicate with synapses and neurons, thereby regulating synaptic activity [4]. The astrocyte cells together with spiking neurons form a Spiking Astrocyte-Neural Network (SANN) with a distributed and fine-grained self-repair capability.

Neural activity levels in the nervous system rely on various plasticity mechanisms, environmental variations and developmental changes. Synaptic plasticity is a mechanism used by neurons to counteract excessive excitation or inhibition by adjusting synaptic strengths [5], [6]. In this work, we propose such a synaptic plasticity mechanism in hardware where the activity of a post synaptic neuron is used to modify synaptic weights to overcome faults in the system through learning mechanisms.

Hebbian learning is a synaptic plasticity rule where a synapse between two neurons is strengthened when its pre and post synaptic neurons have highly correlated outputs. Spike Timing Dependent Plasticity (STDP) is an asymmetric form of Hebbian learning induced by tight temporal correlations between the presynaptic and postsynaptic neuron spikes [7], [8]. The Bienenstock, Cooper, and Munro (BCM) synaptic modification rule modulates the postsynaptic activity if it deviates from the required response [9], [10]. In this paper, we use a self-repairing learning rule [11] that uses evidence [12] to explain how the STDP and BCM learning rules co-exist to give a learning function that is under the control of postsynaptic neuron activity.

When faults occur in the pathways between neurons the postsynaptic neurons can fall silent. Silent/near silent neuron presents a weak node in the system. This issue might be due to hardware failures in the system. The proposed work is a solution for faults leading to failures in the interconnections between the neurons. These may be sensor failures, SEUs, stuck-at fault, interconnect fracture, noise, etc. We define repair as the ability of the system to restore firing rates through synaptic weight modulations regulated by learning rules.

To summarize, this paper consists of the following contributions: Firstly, we propose a novel learning rule utilizing STDP and BCM rules, implemented on an FPGA. Secondly, an efficient fault tolerance is demonstrated on the proposed architecture, showing repair capability if at least one healthy synapse exists in the system. Finally, we use the strengths of biological SANN systems with their parallel processing and learning capabilities to solve a real world task.

The rest of the paper is organized as follows. Section II describes the biological background of the self-repairing learning rule. Section III presents a hardware implementation of the proposed self-repair learning methodology. In section IV, we analyze the repair capability of the proposed system implemented on an FPGA. Section V presents experimental results establishing the effectiveness of the proposed scheme. Section VI demonstrates an application of the proposed learning mechanism. Finally, conclusions and future work are discussed in Section VII.

II. Self-repairing Learning Rule

The network uses the Leaky Integrate and Fire (LIF) [13] neuron model (N_1 N_2, N_3 and N_4 in Figure 1), which is a simplified model of a biological neuron widely used in neuromorphic computing. It represents a good trade-off between computational complexity and biological realism. The representation of a LIF neuron is shown in Equation (1).

$$\tau_{mem}\frac{dv}{dt} = -v(t) + R_{mem}\sum_{i=1}^{m} {I_{syn}}^{i}(t) \tag{1}$$

where τ_{mem}, R_{mem}, v and I_{syn} are the time constant, membrane resistance, membrane potential and current injected by a synapse respectively. m represents the total number of synapses connected to the neuron. On reaching the threshold voltage, the membrane potential is brought back and held at 0V following a nominal refractory period (2 clock cycles). The expression is evaluated using the Euler method.

A tripartite synapse is a junction between a presynaptic terminal, a postsynaptic terminal and an astrocyte cell. Let us consider a signaling pathway between these three terminals and a GABA interneuron terminal. We assume that the presynaptic spiking frequency (f_{pre}) is same as that of the GABA neuron. Interneuron-astrocyte signaling dynamically affects excitatory neurotransmission. Astrocytes, by the action of retrograde signaling [12], regulates transmission of spikes between two layers of the network (A^* in Figure 1). For example, under low f_{pre}, the interneuron-astrocyte interaction leads to an inhibitory effect representing a low transmission probability (PR) with the associated neuron. However, at high f_{pre}, the effect gets reversed leading to a high transmission probability. These outcomes are due to release of inositol 1, 4, 5-trisphosphate (IP_3), Calcium and Glutamate released within the astrocyte [14]. The above phenomenon controls the spikes arriving the postsynaptic terminal by regulating the PR and can be mathematically modeled by Equation (2).

$$PR = exp\left(\frac{(f_{fre} - f_s)^2}{2\sigma^2}\right) \tag{2}$$

In biological systems, dendrites of a presynaptic neuron makes multiple connections with a postsynaptic neuron [15]. The propagation delays between these paths may differ. Hence, in the proposed SANN, each pre-post neural coupling has multiple synaptic pathways (8 in our experiments) where each pathway is assumed to be morphologically modeled by a different delay. This design of multiple pathways permits the post synaptic potential to build up in a realistic way. Also, more pathways gives greater fault tolerance, but the cost is in implementation overhead.

To implement a self-repairing mechanism for the SANN, a learning algorithm needs to be designed. In this approach, the STDP [7], [8], together with BCM learning rule [9], [10] are combined to develop the BCM-STDP rule. STDP uses the time difference between presynaptic and postsynaptic spikes to adjust the synaptic weights, where the equations in (3) cause long-term potentiation (LTP) for $\Delta t \leq 0$ and long-term depression (LTD) for $\Delta t > 0$. In this approach,

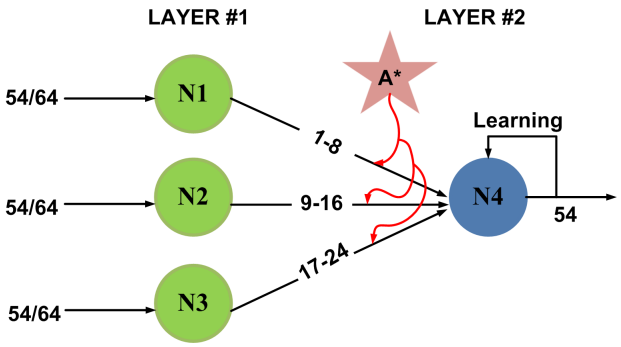

Fig. 1: **Basic unit for fault-tolerant learning mediated by an astrocyte**: Neurons N_1, N_2 and N_3 resides in layer 1 and N_4 resides in layer 2 of the SANN. Astrocyte A^* regulates the PR of inputs received by N_4 from N_1, N_2 and N_3. There are 8 parallel paths (synapses) between each pair of presynaptic and postsynaptic neuron. The input layer receives signals 54 or 64 spikes/window. A^* modulates PR so as to permit the selected pattern to the neuron N_4. Based on the input pattern, the system learns to achieve the required spike rate (54 spikes/window).

potentiation/depression is described by, Equation (3)

$$\delta w(\Delta t) = \begin{cases} +A_0.exp\left(\frac{\Delta t}{\tau_+}\right), \Delta t \leq 0 \\ -A_0.exp\left(\frac{-\Delta t}{\tau_-}\right), \Delta t > 0 \end{cases} \tag{3}$$

where $\delta w(\Delta t)$ is the weight update, A_0 is the time varying height of STDP learning window controlling the maximum levels of weight potentiation and depression, τ_+ and τ_- control the decay rate of weight updating. Δt=($t_{pre} - t_{post}$) is the time difference between presynaptic spike time (t_{pre}) and postsynaptic spike time (t_{post}). Symmetrical plasticity window is assumed and τ_+=τ_-=5 clock cycles. In addition, the BCM learning rule modulates the height of the STDP plasticity window as a function of the firing rate. This is modelled by Equation (4).

$$A_0 = \frac{A}{1 + exp\left(a\left(f - f_0\right)\right)} - A_- \tag{4}$$

where f and f_0 are the actual and target firing rate of the postsynaptic neuron (N_4 in Figure 1), respectively. A is the maximum height of plasticity window and A_- is the maximum height of plasticity window for depression. The parameter a is constant which controls the opening/closing speed of plasticity window and is found experimentally to be 0.1. The BCM rule modulates the synaptic plasticity level by A_0, associated with STDP, as a function of postsynaptic neural activity. Specifically when neurons are inactive, either early on in the normal learning phase or when a fault occurs, the window is shifted up the vertical axis (A_0 increases) and as the postsynaptic neuron activity increases, as learning progresses, the plasticity window is moved down the vertical axis (A_0 decreases) until learning ceases.

III. Hardware Implementation of Self-learning Rules

The basic sub-block implemented on the FPGA for detecting a three bit pattern associated with firing rates is shown

in Figure 1. We use a window of size 2^{10} clock cycles to determine the spike rate of the post synaptic neuron. Moving average is used to determine the spike frequency. Input spike trains of frequency 54 spikes/window and 64 spikes/window are used to represent the input pattern. Neurons in layer 1 facilitate transmission of spike trains. There are 8 parallel variable delay paths between the presynaptic and postsynaptic neuron. Parallel paths allow the postsynaptic potential to build up in neuron $N4$. Astrocyte A^* regulates the spikes received by the neuron in layer 2. For example, let us consider a case in which the sub-block presented in Figure 1 is used for detecting a pattern $(54, 54, 64)$. If pattern $(54, 54, 64)$ is selected, the neurons in layer 1 produces spikes of frequencies 54, 54 and 64 respectively. Astrocyte monitors the spikes and generates transmission PR depending on the pattern to permit the selected pattern in the neuron N_4. Considering a compact hardware implementation, the PR represented in equation 2 consumes more resources and should be approximated. We try to implement equation 2 using a set of piecewise linear equations (8 in our implementation). Due to the astrocyte PR regulation, If the pattern $(54, 54, 64)$ is received by the input neurons, the spike trains are delivered to the neuron N_4. Astrocyte restricts any other patterns of firing rates to be transited to N_4 by lowering PR as per equation (2).

The neurons in the system are LIF and have the following parameters. $R_{mem} = 1M\Omega$, V_{th} (threshold voltage) = $15mV$, τ_{mem}=$10ms$ and resting membrane potential V_r = $0v$. Euler method of integration evaluates the LIF expresion with a fixed time step of $\Delta t = 2^{-10}$s (an approximation for 1ms). The synapses used in this model are probabilistic in nature. We use a uniformly distributed pseudorandom number generator to produce a random number which is denoted by $rand$. We use a Linear Feedback Shift Register (LFSR) to generate $rand$. If $rand$ is less than or equal to the release PR, synapse injects a current to the neuron.

$$I_{syn} = \begin{cases} I_{inj}, rand \leq PR \\ 0, rand > PR \end{cases} \tag{5}$$

Based on the input pattern, neuron N_4 learns to achieve the required spike rate (54 spikes/window). Learning is achieved using STDP and BCM rules. As per equation 3 and equation 4, if output frequency deviates from the required output frequency (54 spikes/window for neuron N_4), the weights of synapses are updated by a certain amount. Equation 3 is approximated using the following relation.

$$\delta w^*(\Delta t) = \begin{cases} +A_0.2^{\left(\frac{\Delta t}{\tau_+}\right)}, \Delta t \leq 0 \\ -A_0.2^{\left(\frac{-\Delta t}{\tau_-}\right)}, \Delta t > 0 \end{cases} \tag{6}$$

Equation 6 is efficient for hardware implementation as this requires A_0 to be shifted by a certain amount controlled by Δt. Considering a compact hardware implementation, the BCM rule represented in equation 4 consumes more resources and should be approximated. We implement equation 4 using a set of 8 piecewise linear equations. Based on the synaptic weights, a current is generated.

$$I_{inj} = (w + \delta w)^*.\varepsilon \tag{7}$$

where w is the weight of the synapse at the current time step and ε is a scaling factor determined empirically. In our implementation $\varepsilon = 2^{-6}$.

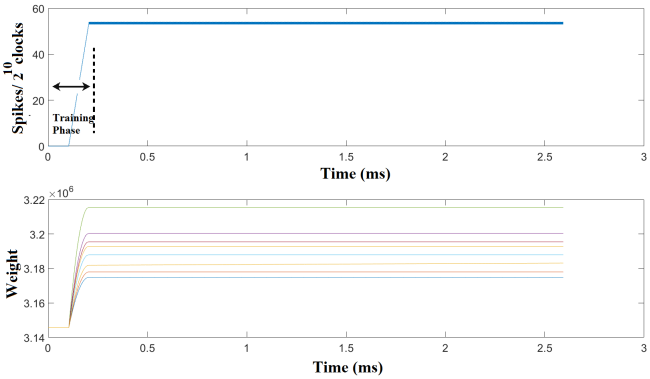

Fig. 2: **(A) Firing rate of N_4 under fault free condition**: All synapses between neurons in layer 1 and layer 2 are fault free enabling 8 parallel delayed paths between each connected neuron. A stable firing rate of 54 $spikes/window$ is established around $0.25ms$. **(B) Synaptic weights under fault free condition**: During the training phase, the synaptic weights (24 plots: 8 synapses $\times 3$) increases and achieves a stable value around $0.25ms$.

One aspect of our model is that it operates at an accelerated biological time scale similar to that in [16], proving to be an efficient realization of real-world tasks compared to [11].

IV. Fault Analysis

A. No Faults

Three input spike trains of frequency 54 spikes/window, 54 spikes/window, and 64 spikes/window are used for testing the basic functionality of architecture depicted in Figure 1. These input spike trains are compatible with the center frequency of the associated synapses and hence the astrocyte delivers a high PR for this pattern. The target frequency of neuron N_4 is set to be 54 spikes/window. A normal learning phase will occur and the spike rate gradually increases during the training phase and eventually stabilizes at the target frequency of 54 spikes/window in $0.25ms$ as shown in Figure 2-A. Figure 2-B shows the synaptic weights which, as expected, show a slow rise over the learning period and stabilize at $0.25ms$. Additionally, the system is tested with presynaptic spike train frequencies outside the filter window (patterns other than (54,54,64) spikes/window) (results not shown) showed that no learning occurred, which verified that the proposed network is selective to the input spike train patterns.

B. Partial Faults

To evaluate the self-repairing capability of the proposed SANN, the spike train frequency of both N_1, N_2, and N_3 was set to the center frequency of the Gaussian PR curve (54,54,64 spikes/window respectively) and the SANN was trained with a target frequency for N_4 of 54 spikes/window. Next, the system is subjected to faults gradually after every $100ms$ (Figure 3). For example, at $100ms$, one of the path between N_1 and N_4 is broken, we can see an activity drop here in Figure 3-A, which subsequently causes the learning window to re-open and the training process restarts. Likewise, we gradually increase the

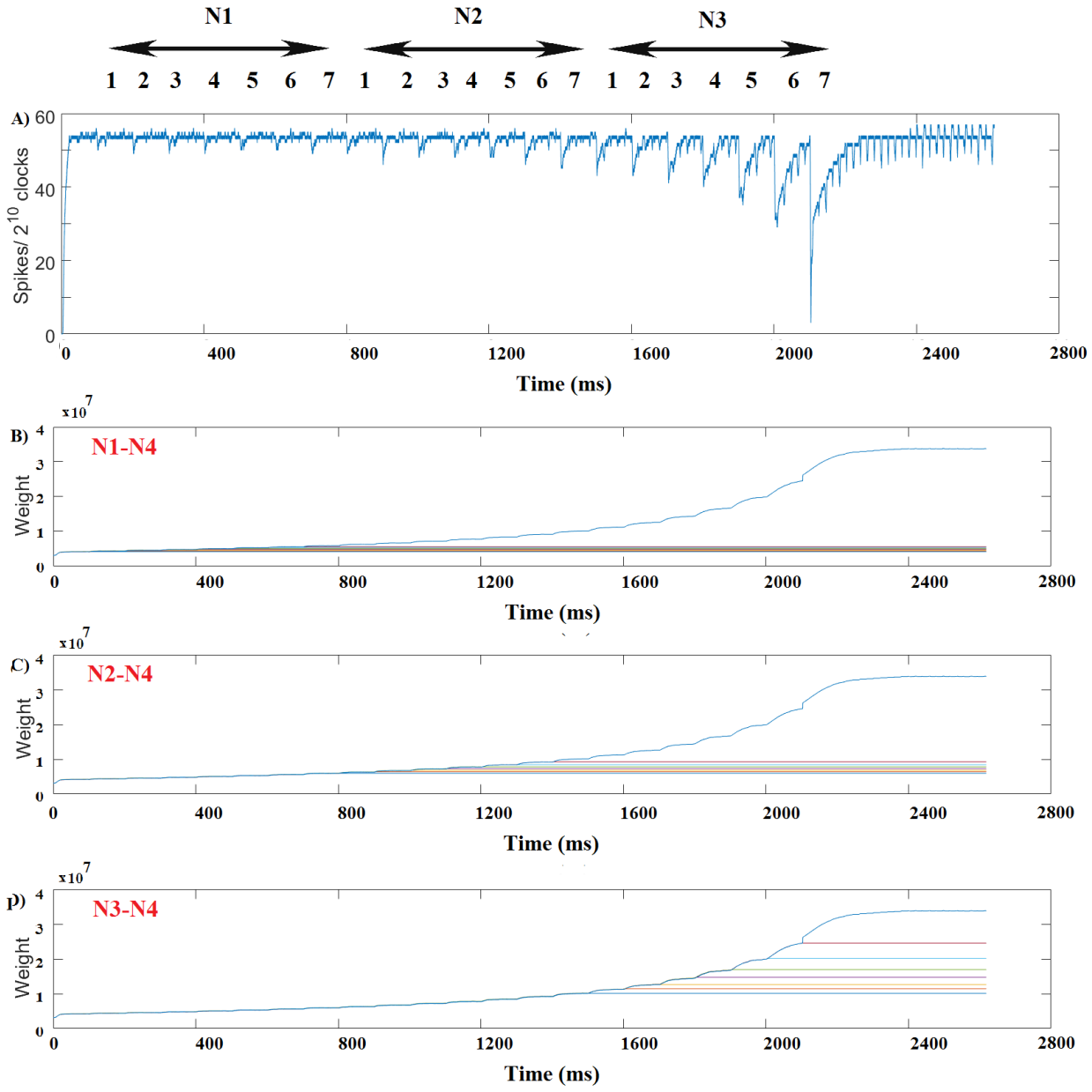

Fig. 3: **(A) Firing rate of N_4 under different faulty conditions**: Synapses between neurons in layer 1 and layer 2 are induced faults gradually (1-7 under neuron N_1 specifies the point in which each synapses between N_1 and N_4 is broken). A stable firing rate of $54 spikes/window$ is established in all faulty cases. **(B) Synaptic weights between N_1 and N_4 under faults of different percentage**: We induce gradual faults on the system after every $100ms$. Only one synapse of N_1 is left unbroken after $700ms$. **(C) Synaptic weights between N_2 and N_4 under faults of different percentage**: We induce gradual faults on the system after $700ms$. The only synapse of N_2 is left unbroken after $1400ms$. **(D) Synaptic weights between N_3 and N_4 under faults of different percentage**: We induce gradual faults on the system after $1400ms$. Only synapse of N_2 is left unbroken after $2100ms$. We can see that for each fault percentages, the systems learn and modulates the synaptic weights for establishing a constant firing rate similar to the fault-free results. The broken synapses do not increase the weight it just retains the weight.

faults in synapses between N_1 and N_4 leaving at least one (No repair happens if all paths between N_1 and N_4 break). At $700ms$, 7 synapses between N_1 and N_4 are broken. Then we induced faults in synapses between N_2 and N_4. At $1400ms$, 7 synapses between N_2 and N_4 are broken and at $2100ms$, 7 synapses between N_3 and N_4 are broken. Hence weights of nonfaulty synapses increase to compensate for synaptic input to N_4. This process of re-training the network to recover the firing rate of N_4 defines the self-repairing process. The modulated weights of synapses associated with N_1, N_2, N_3 are shown in Figure 3-B, C and D respectively. It is evident from Figure 3-A that the fault repair happens at a rate of ms, proving

TABLE I: **Hardware Overhead for the basic unit(Figure 1) implemented on the FPGA**

Methodology/Components	Slice	Slice Reg	LUT	BRAM	DSP
8 delayed paths per input	3154	3407	9108	0	128
A_0 Generator	140	32	470	0	16
Moving Average	11	42	42	2	0
LIF Neuron	7	36	78	0	7

to be a very efficient scheme for real world applications. This is a very important capability for the fault-tolerant hardware systems due to the distributed, fine-grained repair capability which will yield a significantly enhanced performance over conventional approaches [17], [18].This enhancement is also due to ability of the system to run at an accelerated time scale.

V. Hardware results

The proposed architecture is implemented on a Xilinx Virtex-7 FPGA board. The firing rates of the output layer neurons are monitored using the *Xilinx ChipScope Pro* analyzer. Power estimation of the circuits was carried out using *Xilinx XPower Analyzer* and timing analysis using *Xilinx Timing Analyzer*. Table I reports the hardware resource footprint of the proposed models. Total on-chip power dissipation of the system is $0.535W$. As evident from these reports, the proposed architecture of neural self-learning can be implemented on the FPGA with minimal hardware overhead and power consumption. But, on considering large network architectures and applications, the proposed implementation should be improved in terms of hardware especially in the amount of DSP consumption. In our implementation, we have 8 equations for realizing PR/synapse and 8 equations for realizing A_0. Each equation consumes 2 DSPs, proving to be a constrain for larger SANNs. We are considering alternative approximation techniques to overcome this issue. The results presented in Figure 2 and Figure 3 corresponds to the system operating at a frequency of $10MHz$. The maximum operating frequency observed for the complete system is $60MHz$. Hence fault recovery can happen 6 times faster than the reported figures.

VI. Application

The main aim of the proposed work is to implement fault tolerant SANN in FPGA. Once this is established successfully, we apply the concept to a real-world task. There are some works which demonstrate the application of FPGA-based neural networks in solving real world tasks [19]–[21]. Compared to these works, the proposed system has enhanced fault-tolerance and learning capabilities. The emphasis of the selected application is on the fault recovery while establishing the task of robot navigation towards the direction of a colored target (say red). The task implemented by the proposed architecture can be explained in association with Table II. The image of the red target is sensed in forward, right and left direction. Based on the direction of the red target either in forward, right or left direction, the FPGA input pins reflect a value of logic 1 in F_c, R_c or L_c respectively. The system receives information on the presence of an obstacle and is presented a logic 1 in F_o, R_o or L_o on pins of the FPGA in the forward, right or left directions respectively. D represents the final direction of robot movement. In our experiment forward direction has the highest priority and the reverse direction has

TABLE II: **Mapping of input sensor reading to output spikes**

F_c	F_o	R_c	R_o	L_c	L_o	$Direction\ (D)$
0	**0**	**x**	**x**	**x**	**x**	**Forward**
1	**0**	**x**	**x**	**x**	**x**	**Forward**
1	**1**	**x**	**x**	**x**	**x**	**Forward**
0	**1**	**0**	**0**	**x**	**x**	**Right**
0	**1**	**1**	**0**	**x**	**x**	**Right**
0	**1**	**1**	**1**	**x**	**x**	**Right**
0	**1**	**0**	**1**	**0**	**0**	**Left**
0	**1**	**0**	**1**	**1**	**0**	**Left**
0	**1**	**0**	**1**	**1**	**1**	**Left**
0	**1**	**0**	**1**	**0**	**1**	**Reverse**

F_c=1 represents presence of the selected coloured target in the robot's forward direction. F_c=0 represents absence of selected coloured target in the robot's forward direction. F_o=1 represents an obstacle in robots forward direction. F_o=0 represents no obstacle in robots forward direction. F_c=1 and F_o=1 represents an obstacle in robots forward direction, the obstacle is the selected coloured target. Similar meaning holds of the directions Right (R) and Left (L)

the lowest priority (Forward $>$ Right $>$ Left $>$ Reverse). To analyze this let us consider the first case reported in Table II. Here F_c and F_o have a logic 0, since there is no obstacle in the forward direction spikes are delivered in the forward neuron establishing movement in the forward direction. In the second case F_c, F_o=(1, 0) detects the presence of the colored target in the forward direction, hence robot navigates to forward direction. In the third case, we have F_c, F_o=(1, 1), detecting the red target and the obstacle. We assume that the red target is treated as the obstacle and hence the robot navigates towards the red target by having spikes in forward direction. Case F_c, F_o=(0, 1) depicts an obstacle in the forward direction which is not the red target, hence priority goes to the right direction. Similar interpretation holds for the right, left, and reverse directions.

Figure 4 represents the complete system required for navigation considering the specification in Table II. The SANN consists of three layers. The input layer, the hidden layer, and an output layer. The system consists of 6 neurons in the input layer, each detecting either the presence of a red target or an obstacle. These neurons are connected to the hidden layer neuron using 8 variable delay synapses. An astrocyte provides required PR tuning for each pattern and enables the encoded pattern to be passed to the hidden layer neuron. For example, there are 10 patterns to be distinguished by the hidden layer and hence there are 10 astrocytes (not shown in the figure) and 10 hidden layer neurons for detecting these patterns. If there is a spike in the forward direction, this disables spike generation in right, left and reverse direction. The output layer neuron combines the hidden layer neurons responsible for each direction. Since we consider 4 directions, we have 4 motor neurons in the output layer.

VII. Conclusion

In this paper, we discussed three contributions. Firstly, we built a fault tolerant neuromorphic architecture for SANNs. Based on this self-repairing mechanism, when faults occur and the synaptic connection is broken, the network still retains the capability to reorganize itself by re-training and consequently recover to the pre-fault mapping. Secondly, we discussed an approximate model of the system for FPGA-based implementations. The work presented represents an initial step towards a new form of fault tolerant designs, with low overhead and

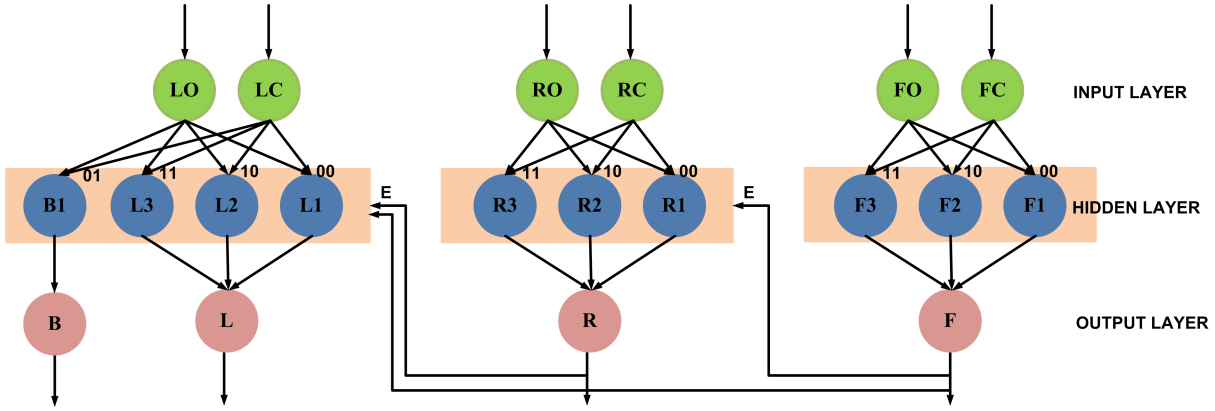

Fig. 4: **Application establishing robot navigation task** There are 6 input layer neurons, each receives information on the presence of a colored target or obstacle. Astrocytes enable specific patterns to be delivered to the hidden layer neuron (Astrocyte is not shown in the figure). There are 4 motor neurons in the output layer, each delivering directionality information for particular directions. The priority of movements is (Forward > Right > Left > Reverse), which is established using binary enable signals (E).

high performance. We are working further towards a more compact architecture based on the proposed methodology for large scale implementations of SANNs. Finally, the proposed idea is applied to a robotic application. The proposed architecture is appropriate for FPGA-based applications running in environments that induce faults in systems, where reliability is crucial.

VIII. Acknowledgements

The work is part of the SPANNER project and is funded by EPSRC grant(EP/N007050/1, EP/N00714X/1). Additionally, the authors would like to acknowledge the platform grant(EP/K040820/1) funded by EPSRC.

References

[1] C. Mead and M. Ismail, *Analog VLSI implementation of neural systems*. Springer Science and Business Media, 2012, vol. 80.

[2] A. Cassidy, S. Denham, P. Kanold, and A. Andreou, "FPGA Based Silicon Spiking Neural Array," in *IEEE Biomedical Circuits and Systems Conference*, Nov. 2007, pp. 75–78.

[3] L. E. Clarke and B. A. Barres, "Emerging Roles of Astrocytes in Neural Circuit Development," *Nature Reviews Neuroscience*, vol. 14, no. 5, pp. 311–321, 2013.

[4] B. Stevens, "Neuron-astrocyte Signaling in the Development and Plasticity of Neural Circuits," *Neurosignals*, vol. 16, no. 4, pp. 278–288, 2008.

[5] G. G. Turrigiano, "The self-tuning neuron: Synaptic Scaling of Excitatory Synapses," *Cell*, vol. 135, no. 3, pp. 422–435, 2008.

[6] K. Pozo and Y. Goda, "Unraveling Mechanisms of Homeostatic Synaptic Plasticity," *Neuron*, vol. 66, no. 3, pp. 337–351, 2010.

[7] L. F. Abbott and S. B. Nelson, "Synaptic plasticity: taming the beast," *Nature neuroscience*, vol. 3, pp. 1178–1183, 2000.

[8] S. Song, K. D. Miller, and L. F. Abbott, "Competitive Hebbian Learning through Spike-Timing-Dependent Synaptic Plasticity," *Nature neuroscience*, vol. 3, no. 9, pp. 919–926, 2000.

[9] E. L. Bienenstock, L. N. Cooper, and P. W. Munro, "Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex," *Journal of Neuroscience*, vol. 2, no. 1, pp. 32–48, 1982.

[10] M. Bear and F. Ebner, "A physiological basis for a theory of synapse modification," *WORLD SCIENTIFIC SERIES IN 20TH CENTURY PHYSICS*, vol. 10, pp. 121–130, 1995.

[11] J. Liu, L. McDaid, J. Harkin, J. Wade, S. Karim, A. P. Johnson, A. G. Millard, D. M. Halliday, A. M. Tyrrell, and J. Timmis, "Self-Repairing Learning Rule for Spiking Astrocyte-Neuron Networks (accepted)," in *Proceedings of the 9th International Conference on Neural Information Processing (ICONIP)*, 2017.

[12] 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," *Frontiers in computational neuroscience*, vol. 6, 2012.

[13] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge university press, 2002.

[14] G. Perea, R. Gómez, S. Mederos, A. Covelo, J. J. Ballesteros, L. Schlosser, A. Hernndez-Vivanco, M. Martín-Fernndez, R. Quintana, A. Rayan, A. Díez, M. Fuenzalida, A. Agarwal, D. E. Bergles, B. Bettler, D. Manahan-Vaughan, E. D. Martín, F. Kirchhoff, and A. Araque, "Activity-dependent Switch of GABAergic Inhibition into Glutamatergic Excitation in Astrocyte-neuron Networks," *Elife*, vol. 5, pp. 1–26, Dec. 2016.

[15] J. Smith and M. Martonosi, *Space-Time Computing with Temporal Neural Networks*, ser. Synthesis Lectures on Computer Architecture. Morgan & Claypool Publishers, 2017.

[16] N. Jing, J.-Y. Lee, Z. Feng, W. He, Z. Mao, and L. He, "SEU Fault Evaluation and Characteristics for SRAM-based FPGA Architectures and Synthesis Algorithms," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 18, no. 1, p. 13, 2013.

[17] A. P. Johnson, D. M. Halliday, A. G. Millard, A. M. Tyrrell, J. Timmis, J. Liu, J. Harkin, L. McDaid, and S. Karim, "An FPGA-based hardware-efficient fault-tolerant astrocyte-neuron network," in *IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2016, pp. 1–8.

[18] J. Liu, J. Harkin, L. P. Maguire, L. J. McDaid, and J. J. Wade, "SPANNER: A Self-Repairing Spiking Neural Network Hardware Architecture," *IEEE Transactions on Neural Networks and Learning Systems*, 2017.

[19] A. P. Johnson, J. Liu, A. G. Millard, S. Karim, A. M. Tyrrell, J. Harkin, J. Timmis, L. J. McDaid, and D. M. Halliday, "Homeostatic Fault Tolerance in Spiking Neural Networks: A Dynamic Hardware Perspective," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. PP, no. 99, pp. 1–13, Jul. 2017.

[20] A. P. Johnson, J. Liu, A. G. Millard, S. Karim, A. M. Tyrrell, J. Harkin, J. Timmis, L. McDaid, and D. M. Halliday, "Homeostatic Fault Tolerance in Spiking Neural Networks utilizing Dynamic Partial Reconfiguration of FPGAs (accepted)," in *2017 International Conference on Field-Programmable Technology (FPT)*, Dec. 2017.

[21] J. B. George, G. M. Abraham, B. Amrutur, and S. K. Sikdar, "Robot Navigation Using Neuro-electronic Hybrid Systems," in *28th International Conference on VLSI Design*, Jan. 2015, pp. 93–98.