Case Study: Bio-inspired Self-adaptive Strategy for Spike-based PID Controller

Junxiu Liu, Jim Harkin, Malachy McElholm, Liam McDaid
School of Computing and Intelligent Systems,
University of Ulster, Magee Campus,
Derry, Northern Ireland, UK
{liu-j4, jg.harkin, m.mcelholm, lj.mcdaid}@ulster.ac.uk

A. Jimenez-Fernandez, A. Linares-Barranco
Robotic and Technology of Computers Lab,
University of Seville,
Seville, Spain
{ajimenez, alinares}@atc.us.es

***Abstract*— A key requirement for modern large scale neuromorphic systems is the ability to detect and diagnose faults and to explore self-correction strategies. In particular, to perform this under area-constraints which meet scalability requirements of large neuromorphic systems. A bio-inspired online fault detection and self-correction mechanism for neuro-inspired PID controllers is presented in this paper. This strategy employs a fault detection unit for online testing of the PID controller; uses a fault detection manager to perform the detection procedure across multiple controllers, and a controller selection mechanism to select an available fault-free controller to provide a corrective step in restoring system functionality. The novelty of the proposed work is that the fault detection method, using synapse models with excitatory and inhibitory responses, is applied to a robotic spike-based PID controller. The results are presented for robotic motor controllers and show that the proposed bio-inspired self-detection and self-correction strategy can detect faults and re-allocate resources to restore the controller's functionality. In particular, the case study demonstrates the compactness (~1.4% area overhead) of the fault detection mechanism for large scale robotic controllers.**

Keywords*— *Bio-inspired system, robotics, fault tolerant, self-correction, hardware adaption.

I. INTRODUCTION

Bio-inspired systems provide high levels of parallel computing which model varied levels of computationally efficient biological systems. Various approaches have been proposed in this field, e.g. neuromorphic chip [1], bio-inspired systems of visual object recognition and tracking [2], and neuro-inspired controller [3]. Bio-inspired computing uses a set of layers to process complex input data and produces predicted results. The layers include sensors input, data processing and learning layers. For example, the neuromorphic chip (i.e. 'brain') of a bio-inspired robot receives data from external sensors. The data is processed with other cognitive and history information (i.e. 'learning') and decisions are made to execute corresponding movements (i.e. 'body') of the robot.

In our previous work [3], a spike-based Proportional-Integrative-Derivative (PID) robot controller was proposed which employed the last layer of a spiking neural network (SNN) to control a motor. The controller converts motor sensor signals into spikes and feedbacks to the SNN to aid in making speed-control decisions. However, a robot generally requires multiple motor controllers to control the motor movements. The implementation of these controllers requires significant hardware resources. Since modern neuromorphic systems are designed and implemented based on large-scale high density chips, it is becoming increasingly challenging to guarantee that the system will continuously operate fault-free. As device geometries scale down, the reliability of devices become lower, exhibiting higher risks of faults occurring post-manufacturing. Faults in electronic systems include permanent faults from a wear-out effect, rupture in a manufacturing process and temporary faults caused by power supply fluctuations and radiation effects. Fault models at the gate level are mostly stuck-at, bridging, crosstalk faults etc. In neuromorphic systems such faults cause the spike transmission and processing to become corrupted leading to failure in operation. Therefore, the presence of faults requires neuromorphic systems to be adaptive to faulty conditions post-manufacturing. A key fundamental task in realising robust or adaptive neuromorphic system is firstly the detection of faults (including permanent and temporary faults) and then the application of a correcting strategy to restore functionality. A major challenge is the investigation of a self-detecting and self-correcting architecture that can offer online fault detection and correcting after faults occur, without significant increasing of cost in terms of area overhead and power consumption.

This paper presents a case study into a self-detection and self-correcting strategy with a compact area overhead for the spike-based PID controller. The strategy uses a bio-inspired synapses and neurons to detect spike anomalies, and therefore faults, by mimicking the fault tolerance and self-adaptive capabilities of the biological neural system. These capabilities are then applied to the robotic controller to improve robustness. The rest of this paper is organised as follows: section II outlines the spike-based PID controller and bio-inspired fault detection strategy, and section III discusses the proposed fault detection and self-correcting case study for the PID controller application. Section IV presents results and performance analysis and section V provides a conclusion and highlights the future work.

II. BACKGROUND AND PREVIOUS WORK

A spike-based PID controller was proposed in our previous work [3] and its structure is shown in Fig. 1. The controller includes the following modules: Synthetic Spikes Generator, spikes Hold and Fire (H&F), Spikes Integrate & Generate (SI&G), Spike Temporal Derivate (STD), Spikes Expander (SE) and Quadrature encoder to Spike Rate (QSR). The controller uses the spike stream to control the motor rotation

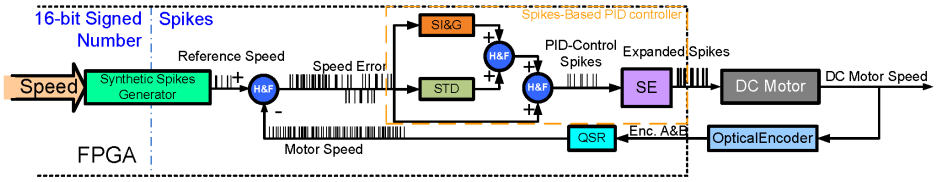


Fig. 1. Spike-based PID controller [3]

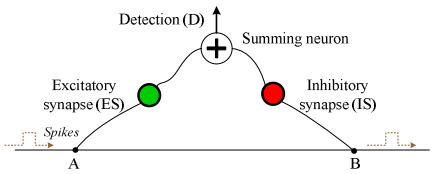


Fig. 2. FDU applied to a single wire [4]

for a given fixed speed and it can be replicated in parallel for the multiple motors control in the robot due to its low hardware overhead [3]. A bio-inspired online fault detection strategy for Networks-on-Chip interconnect, namely SMART, was proposed in our previous work [4]. It employs a Fault Detection Unit (FDU) which uses synapse models with excitatory and inhibitory responses as a method for fault detection. Fig. 2 illustrates the signal connections of the FDU for a single wire or connection, where the beginning point A is connected to an excitatory synapse (ES) and the end point B is connected to an inhibitory synapse (IS), with the output of both synapses fed to a summing neuron to generate the detection result (D). If the wire is fault-free, the spike experienced at point A initiates the ES to produce a excitatory response, while at point B the spike initiates the IS to produce an inhibitory response: since there is no temporal delay the spike at A and B occur at the same time. Therefore, the sum of the two synapses is below a defined detection threshold and the output D is constant. However, if a fault (temporal delay) occurs in the line A to B, then the summed value exceeds the threshold, a spike train output from D will be generated indicating a fault is detected. This information is fed to the self-correcting strategy to make a decision on how to accommodate the fault.

There are only spikes flowing between the modules inside the spike-based PID controller [3]. Therefore, the FDU can be applied to the PID controller for the fault detection which outputs the pulses if faults occur. The remainder of this paper will summarise some findings to date on the success of detecting when the PID controller is faulty and what mechanisms we can use to activate additional fault-free controllers to take over. The novelty of this work resides in how to detect such faults and initiate corrective action, and its application in providing robust robotic control.

III. FAULT DETECTION AND SELF-CORRECTING STRATEGY

In this section, the fault detection strategies of single and multiple spike-based PID controllers are presented. The self-correcting strategy under detection of faults is also given.

A. Fault detection for spike-based PID controller

In [3], instead of using Pulse Width Modulation, the Pulse Frequency Modulation is employed to drive the DC motor. The DC motor power is controlled by the frequency of the spikes as shown by equation (1),

$$V \approx T_h * S_r * V_{PS} \quad (1)$$

where V is the DC motor power, T_h is the fixed modulated signal high time, S_r is the spike rate (input) and V_{PS} is the DC voltage provided by the power supply. Therefore, the spike rate of the input reference speed can be controlled to adjust the voltage of the motor (i.e. speed). Motor speed is also measured by the spikes where a QSR module is used to convert the quadrature encoder into the spikes [5].

A Leaky Integrate and Fire neuron is used to model the neuron in the FDU. The membrane potential of the Leaky Integrate and Fire model is presented by (2),

$$\tau_m \frac{dv}{dt} = -v(t) + R_m I_{tot}(t) \quad (2)$$

where v is the membrane potential, R_m is the membrane resistance, I_{tot} is the overall current generated by the synapses connected to the neuron, τ_m is the time constant of the membrane. The neuron does not generate a spike when below the threshold; therefore a variable threshold is required, as shown by (3),

$$Vth_{new} = m \times Vth_{orig} \times \exp\left(-\frac{t-t_i}{\tau_{decay}}\right) \quad (3)$$

where Vth_{orig} is the original firing threshold, t is the current time, t_i is the time a spike is generated, τ_{decay} is the rate at which the threshold decays back to original values and m is the multiplication factor to calculate the new threshold - Vth_{new}. Fig. 3 shows when the membrane potential does not exceed the threshold and therefore spike is not output. Both the ES and IS potentials are summed to provide the neuron membrane potential. Therefore, as they are equal but opposite in magnitude they cancel, thus the firing threshold is not reached. Fig. 4 shows when the ES potential dominates the IS, the firing membrane threshold is reached and a spike is output.

As the input (i.e. reference speed) and output (i.e. motor speed) of the controllers are spikes, they can be used to initiate the ES and IS to produce the response and then to determine

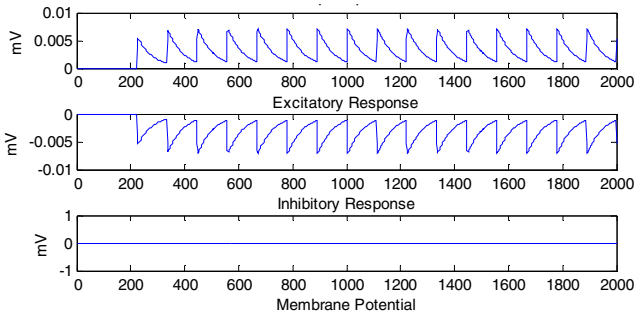


Fig. 3. Firing threshold of membrane potential not reached

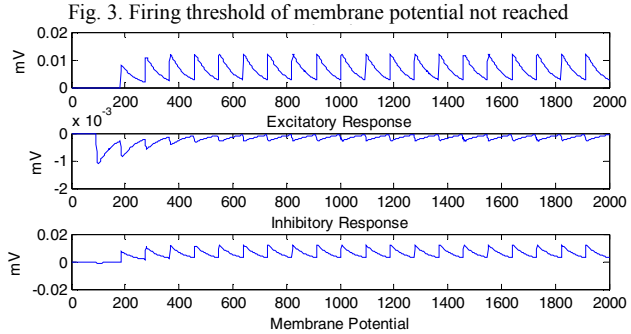


Fig. 4. Firing threshold of membrane potential reached

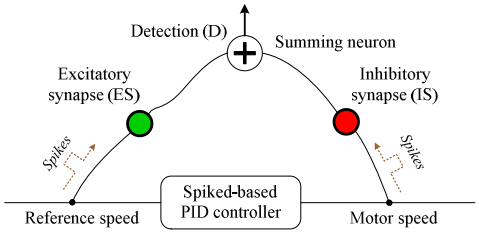
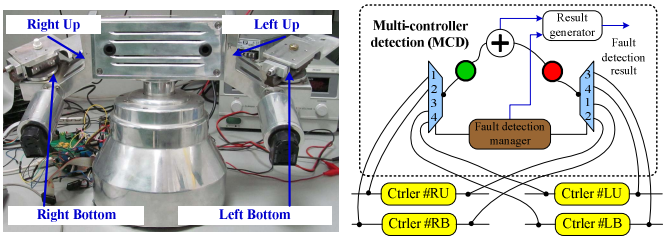


Fig. 5. FDU applied to spike-based PID controller



(a). A robot with four motor controllers (b). Fault detection mechanism for multiple controllers

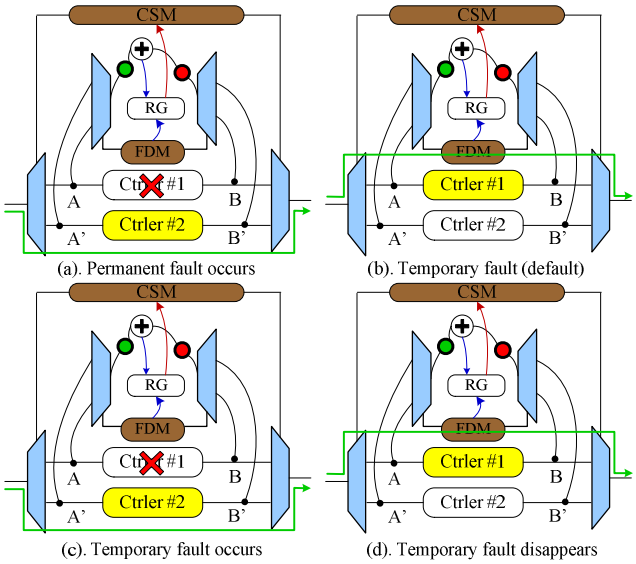
Fig. 6. Fault detection of multiple PID controllers

whether the system is faulty according to whether the neuron spikes. Fig. 5 illustrates the fault detection mechanism for the controller. The reference speed (reflected in the spike activity) activates the ES and the motor speed, again reflected in spike activity, activates the IS. If the controller is fault-free, the sum of two synapse responses at the summing neuron (i.e. the membrane potential) will be below the detection threshold and therefore no fault will be present; when above the threshold a fault is detected. The detected fault signal indicates that a fault occurs in the PID controller, DC motor or optical encoder. The fault location can be determined after combining detection results from other components.

A robot generally requires multiple motor controllers. Fig. 6(a) is an example LINCE robot where four PID controllers are used to control the rotation of the 'eyes' and base. The controller positions of these motors are labelled as Right Up/Bottom, Left Up/Bottom, respectively. The fault detection mechanism of the multiple controllers is presented in Fig. 6(b). It can be seen that the fault detection manager (FDM) selects the controller under test. While the controller is under test, its input and output will be connected to the FDU. The fault detection result and the controller number will be outputted to the result generator (RG) module which generates the final detection result. The final result provides a diagnosis of the fault type and controller location, for example, it includes the controller position and status, e.g. the left up controller is fault-free, left bottom has a permanent fault etc. Note that the detection sequence in the FDM can be set to be first come first served or priority based depending on the application. The advantages of this detection mechanism are that only one FDU is required for the fault detection of multiple controllers and the area overhead is efficient due to the compact design of the FDM and RG. Alternatively, instead of time-division multiplexing, we can also use one individual FDU for each controller as the area overhead of an FDU is much lower than the controller (will be demonstrated in section IV).

B. Self-correcting strategy

A self-correcting strategy for one single motor is illustrated in Fig. 7. The example strategy uses a redundancy model where two PID controllers are available, one is the default and the other is the redundant (i.e. spare). In this approach two FDUs and one Controller Selection Module (CSM) are used in the system. The status outputs (i.e. spiking output) of the two controllers are connected to the CSM. The PID controller has an FDU which monitors its I/O activity for irregular activity (potential faults). In Fig. 7(a) PID controller #1 is used as the default and operational on power-up. However, if faults occur in controller #1, the FDU detects it and CSM then selects controller #2 and routes the input and output paths accordingly via the (de)multiplexer to restore system function using #2. This approach enables faults to be detected and then a corrective measure to be taken to restore operation. Note that FDU #1 and #2 are the same physical units when using the time-division fault detection mechanism shown in section III.A. Fig. 7 also presents an example of corrective actions for a temporary fault. It can be seen that if the controller experiences a temporary fault as shown in Fig. 7(b) then during the time period of the fault, the self-correcting strategy will use controller #2 as shown in Fig. 7(c). As controller #1 is default, the FDM checks its condition regularly. If the fault remains, controller #2 is used; if the fault disappears, then controller #1 will be selected as shown in Fig. 7(d).



(a). Permanent fault occurs (b). Temporary fault (default)

(c). Temporary fault occurs (d). Temporary fault disappears

Fig. 7. Corrective actions for different faults

The advantages of using self-correcting strategy are (1) it exploits the use of the FDU output to make corrective decision, (2) it adapts available hardware resources to restore original function in the event of a physical impairment; (3) the CSM module is area-efficient which maintains system scalability (see section IV).

IV. ANALYSIS

This section presents the results of experiments on the performance of fault detection and self-correction for the PID controller. Fig. 8 shows the experimental system structure where the computer is connected to a USB2SPI circuit which converts the USB protocol to the Serial Peripheral Interface (SPI) protocol. The PID controllers receive the parameters from computer via SPI. It is connected to a motor H-bridge driver circuit using CNY74-4 optocouplers and L298 ICs. The motor driver circuit controls the DC motor movement and

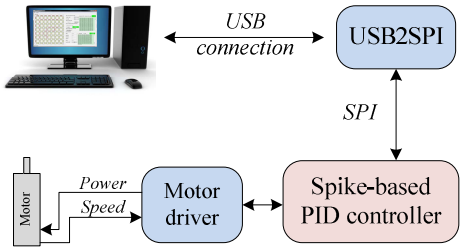


Fig. 8. The experimental system structure

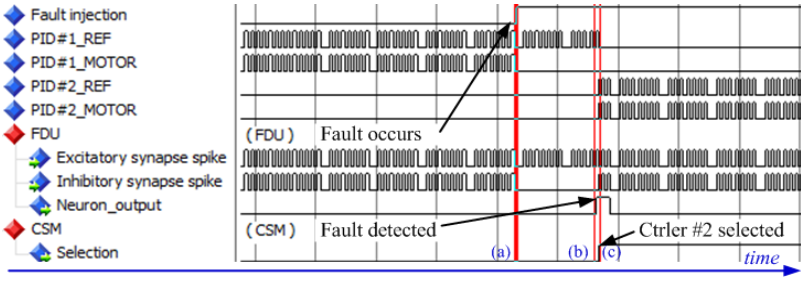


Fig. 9. Self-correcting strategy hardware simulation

receives the speed feedback at the same time. In this work, a system which can control a maximum eight motors is implemented (16 PID controllers are required). Two motor driver circuits are used, i.e. each controls four motors.

A hardware simulation of the fault detection/self-correcting strategy for a single motor is shown in Fig. 9 where the controller #1 is default and #2 is redundant. From power-up to time point (a), controller #1 is active; the reference/motor speed spikes are connected to the ES/IS of FDU; the FDU does not output the spike as controller #1 is fault-free. After time point (a), a permanent fault is injected into controller #1 (fault injection signal is high); the motor speed spike is not generated as the controller #1 is faulty; since the IS of FDU does not receive spikes, after a time period, at time point (b), the FDU outputs a spike which indicates that a fault is detected in controller #1. Therefore at time point (c), CSM selects controller #2 to restore the system function (the selection signal of CSM is high; the reference/motor speed spikes are routed to controller #2). The time periods from point (a) to (b) and (b) to (c), are defined as fault detection (t_d) and self-correction (t_c) periods, respectively. The t_d is 80ms in this work where the strength of synapses in the FDU dictate the speed of detection and also the tolerance between reference and motor spike variations [4]. The t_c is 1 clock cycle (10ns), i.e. after only 1 clock cycle, the system will switch to the redundant controller to restore function promptly.

The area overhead is used to evaluate the compactness of the fault detection and self-correction mechanism. A highly optimised CMOS-based implementation of the excitatory synapse, inhibitory synapse, and integrate and fire neuron have been developed in our previous work [6]. For example, the area consumption of one CMOS-based synapse and one neuron are very low at $24 \times 10^{-8} mm^2$ and $9 \times 10^{-6} mm^2$, respectively. Fig. 5 illustrates that one FDU includes two synapses and one neuron therefore one FDU has a marginal overhead of $9.48 \times 10^{-3} \mu m^2$. Similarly, the CSM, FDM and PID controllers also follow the standard ASIC cell design flow, synthesised and verified based on a SAED 90nm CMOS technology. Table I presents the corresponding area overheads for controlling one motor. It can be seen that the area overhead of the PID controllers is 98.557%. However, the overheads of the proposed fault detection and self-correction mechanism (i.e. CSM, FDM, FDU) is insignificant at 1.443%. These area overheads are very low with the added advantage of fault detection and self-correction. In particular, the approach demonstrates scalability for larger robots where significant numbers of motors are used, e.g. in manufacturing.

TABLE I. AREA UTILIZATION AND OVERHEAD

	Area (μm^2)	Percentage
CSM	0.44	0.531%
FDM	0.75	0.9%
PID Ctrlers	81.75	98.557%
FDU	9.48×10^{-3}	0.012%

V. CONCLUSION

The key aspects of the proposed approach are the principle of fault diagnosis procedure using excitatory and inhibitory synapse models and membrane potential response, fault detection and correction management for multiple PID controllers using FDU, FDM and CSM modules. The results of the case study showed that the proposed mechanism can provide self-detection and corrective capabilities for spiked-based PID motor controllers. In particular, the hardware area overhead is only ~1.4%. Compared to other approaches, the proposed strategies are compact which demonstrated the scalability for large robots containing increased numbers of robotic motors. The future work includes optimizing the fault detection unit parameters to allow a wide range of detection, e.g. explore the strength of synapses; and investigating fault detection mechanisms for other components to aid making decisions about the faults location.

ACKNOWLEDGMENT

This research is supported in part by Spanish grant (with support from the European Regional Development Fund) BIOSENSE (TEC2012-37868-C04-02).

REFERENCES

[1] S. Carrillo, J. Harkin, L. J. McDaid, *et. al*, "Scalable Hierarchical Network-on-Chip Architecture for Spiking Neural Network Hardware Implementations," IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 12, pp. 2451–2461, 2013.

[2] C. Zamarreño-ramos, A. Linares-barranco, and T. Serrano-gotarredona, "Multicasting Mesh AER: A Scalable Assembly Approach for Reconfigurable Neuromorphic Structured AER Systems . Application to ConvNets," IEEE Transactions on Biomedical Circuits and Systems, vol. 7, no. 1, pp. 82–102, 2013.

[3] A. Jimenez-Fernandez, G. Jimenez-Moreno, A. Linares-Barranco, *et. al*, "A Neuro-Inspired Spike-Based PID Motor Controller for Multi-Motor Robots with Low Cost FPGAs," Sensors, vol. 12, no. 4, pp. 3831–3856, Jan. 2012.

[4] M. McElholm, J. Harkin, L. McDaid, and S. Carrillo, "Bio-Inspired Online Fault Detection in NoC Interconnect," in Energy-Efficient Fault-Tolerant Systems, 2014, pp. 241–267.

[5] A. Jimenez-Fernandez, R. Paz-Vicente, M. Rivas, A. Linares-Barranco, G. Jimenz, and A. Civit, "AER-based Robotic Closed-loop Control System," in ISCAS, 2008, pp. 1044–1047.

[6] Y. Chen, L. McDaid, S. Hall, and P. Kelly, "A Programmable Facilitating Synapse Device," in IJCNN, 2008, pp. 1615–1620.