FPGA Implementation of Spiking Neural Networks - an Initial Step towards Building Tangible Collaborative Autonomous Agents

S. Bellis, K. M. Razeeb, C. Saha, K. Delaney, C. O'Mathuna
NMRC, University College Cork, Ireland
sbellis@nmrc.ie

A. Pounds-Cornish, G. de Souza, M. Colley, H. Hagras, G. Clarke, V. Callaghan
University of Essex, UK
apound@essex.ac.uk

C. Argyropoulos, C. Karistianos, G. Nikiforidis
University of Patras, Greece
charg@med.upatras.gr

Abstract

This paper contains the results of an initial study into the FPGA implementation of a spiking neural network. This work was undertaken as a task in a project that aims to design and develop a new kind of tangible Collaborative Autonomous Agent. The project intends to exploit/investigate methods for engineering emergent collective behaviour in large societies of actual miniature agents that can learn and evolve. Such multi-agent systems could be used to detect and collectively repair faults in a variety of applications where it is difficult for humans to gain access, such as fluidic environments found in critical components of material/industrial systems. The initial achievement of implementation of a spiking neural network on a FPGA hardware platform and results of a robotic wall following task are discussed by comparison with software driven robots and simulations.

1. Introduction

Extensive theoretical work has shown that Spiking Neural Networks (SNNs) are deemed computationally more powerful than conventional artificial neural network formalisms [1]. This implies that SNNs need fewer nodes to solve the same problem than conventional artificial neural nets. Relatively little work has been carried out on the implementation of SNNs in digital platforms such as FPGAs [2][3], most work tending to concentrate on analogue ASIC type devices [4][5][6]. FPGA implementation gives the flexibility to develop SNNs for a particular task without committing to costly silicon ASIC fabrication. In addition digitally based SNNs provide a number of other desirable features such as noise-robustness and simple real-world interfaces [7].

The FPGA implementation of the spiking neural network is intended to be the core computational component in what has been dubbed a CAA (Collaborative Autonomous Agent). The work forms part of a European IST project called SOCIAL (Self Organised Societies of Connectionist Intelligent Agents Capable of Learning - IST-2001-38911). The SOCIAL project aims to exploit/investigate methods for engineering emergent collective behaviour in large societies of actual miniature agents that can learn and evolve. It is envisaged that the project will produce tangible agents that can perform individual and collective goal seeking tasks such as repair in environments that are inaccessible for humans. An example of such a task is the repair of bypass tubing used in the oil industry. Pipelines can deteriorate due to scale formation and this can be detected by changes in pH in the vicinity of the fault when the pipes are flushed with water. With integrated pH sensing capability it is intended to build miniature tangible CAAs with the ability to navigate, indirectly communicate amongst themselves, directly communicate with a development environment and possibly repair such faults.

The paper focuses on the initial investigation of the FPGA SNN implementation with the goal of performing a navigational task of wall following. The exercise was used go gain familiarity with SNNs and show that, when implemented on FPGAs, simple navigational tasks can be performed. Initially the SNN used is described in terms of its VHDL implementation resulting from a manual translation of a C-type code implementation of the SNN. The paper will then overview synthesis to FPGA hardware modules used for implementation of the SNN. Interfacing to robotic platforms and results to show comparison with software versions of the SNN on a simulation tool also developed within the project framework will be discussed.

Conclusions will be drawn and the future direction of the project toward implementation of the SNNs on miniature platforms will be projected. New miniature FPGA and communication modules that have been built and proven functional will be presented to show the next phase of implementation.

2. Implementation of SNNs on FPGAs

This section overviews the route taken in the implementation of the SNN controller for the wall following robot on FPGA hardware. A wall following task has been demonstrated on robots running the VxWorks operating system [8]. Essentially this is C-code with extra libraries supporting the real-time robotic operation. It was decided to manually translate this to VHDL as a first pass to FPGA implementation of the SNN. The route to demonstration of the prototype reported in this paper is summarized as follows:

1. Manual translation of the VxWorks C-code to VHDL;
2. Test-bench simulation and comparison with VxWorks C-code output;
3. Synthesis of VHDL to FPGA hardware;
4. Interfacing to a simulation tool developed in the project and the robotic platforms;
5. Test wall following scenario on the simulator and then on the robot.

3. Translation of the C-Code to VHDL

Initially the wall following C-Code was written for the embedded processor of the robots and compiled using VxWorks. The task for VHDL translation involved breaking the code down to its respective functions and representing the entire code in a hierarchical structural description. VHDL entity e_SNN formed the top level of the hierarchy and defines the connectivity between the neurons, weighted synapses and control timer of the network connected as shown in Figure 1.

The network contains two types of neuron. There are two instantiations, front and rear, of the pre-synaptic neurons. On the robots there are ultrasonic sensors located at the front and rear that are used to detect range either from a wall or an obstacle. The two pre-synaptic neurons convert the ultrasonic range values into spike trains, the shape of which defined by a series of exponential functions. These spike trains are weighted in the four synapse entities by a set of weights evolved in software trials on the VxWorks system. The outputs of the synapses feed into the post-synaptic neuron entities. By varying the speed of one motor relative to the other then the robot can turn to avoid an obstacle or follow the path of a wall. The post-synaptic neurons function by firing once a membrane potential has been reached. The time at which they fire has a relationship with the speed of the motor that they are controlling.

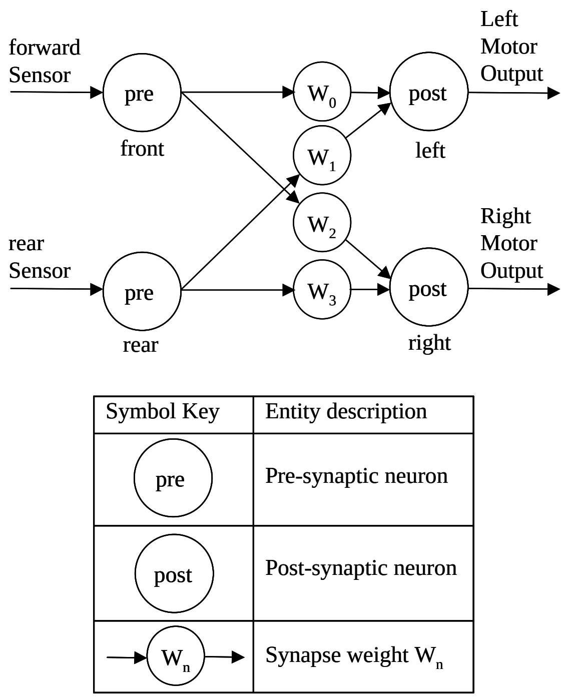

Figure 1. Schematic representation of the VHDL SNN code for entity e_SNN.

In FPGAs it is advantageous to use fixed-point logic so that mathematical functions in the standard libraries can be used to minimize resources. When translating the VxWorks C-Code the precision of signals, sensor inputs and motor outputs had to be decided. Precision directly corresponds to the number of bits used to represent a signal and in turn relates to register storage, logic resources for computation and communication. Optimizing the precision leads to lower resource usage and therefore smaller neurons, ultimately allowing larger SNNs to be implemented on the FPGA. The precision of data within the modules was calculated given the accuracy, noise levels and ranges of the input ultrasonic detectors. The ultrasonic sensors output data in the range 0 to 90 and therefore 7 bits were required to represent this as an unsigned binary number. A five bit signed two’s complement representation was used for the weights.

Overall the VHDL modules contain quite simple arithmetic, concatenation and scaling assignments. Therefore the VHDL standard libraries were used to implement these functions. However, the pre-synaptic neuron block requires more complex arithmetic for the computation of the spike effect. This involves exponential functions that are not contained in the standard VHDL libraries. The exponential function could be broken down into an arithmetical expansion series of calculations and the end result would be computationally expensive in terms of the logic resources on the FPGA. It was therefore decided to model the spike effect as a look-up table in VHDL using case statements.

A C-program was written to generate this look-up table by taking the spike generation exponential functions of the VXWorks C-code and using a decimal to binary conversion routine. The results showed that slight errors were introduced due to quantization of the spike-effect. Note the jagged appearance of the spike effect falling time curve as shown in Figure 2.

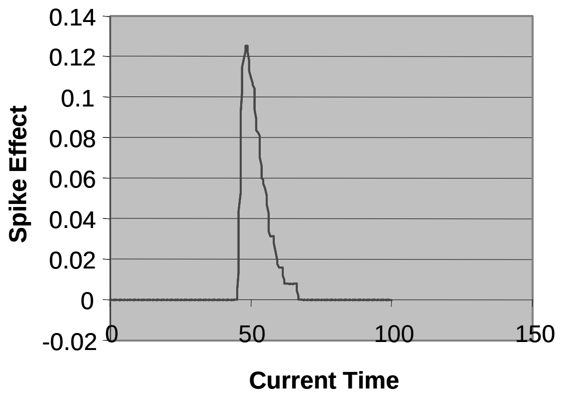

Figure 2. Spike- Effect LUT precision.

4. Synthesis of VHDL to the FPGA

After functional verification against the C-Code wall following model using test-bench methods the synthesis of the VHDL was considered. The FPGA PCB used for the initial testing was originally developed for a research project in which sensor networks were being researched and developed. However a commercially available PCB could equally have been used at this stage. The board uses a Xilinx Spartan II low cost series FPGA, specifically the mid range xc2s300e-7fg456 device [10]. The module has an on board serial configuration EPROM enabling it to be run remotely from a battery source.

The SNN controller was synthesized using Xilinx ISE V6.1 Webpack software. To do this a wrapper VHDL file was created. This wrapper file includes the SNN controller and also port maps the necessary UART components for serial communication with the robot. The implementation of a four-neuron VHDL model of a digital spiking neural network required 324 Slices approximately 10% of the resources on the FPGA device. Therefore approximately 40 neurons would fit on the FPGA for the current version of the neuron. The overall network delay of 8.461ns allows clocking up to a maximum frequency of 118.189MHz. The requirement to clock at 1kHz, based on the specifications of the robots used, to give a 1ms cycle time is therefore met.

5. Interfacing and Hardware Test

A simulation tool is being developed in parallel to the hardware implementation of the SNNs. The simulator allows robot behaviour to be modeled and evolved in software before going to hardware implementation. The simulation tool was also adapted so that it could be interfaced to the FPGA board as shown in Figure 3. By doing so the FPGA implementation could be verified against simulated results to minimize the amount of test and debug needed when interfacing to the real robots. The simulator computes the ultrasonic sensor readings and continuously updates the FPGA via the serial port. The FPGA SNN controller then outputs the motor control data through the serial port and this is used to control the motion of the simulated robot. The FPGA controlling the movement of a virtual robot exhibited the wall following behaviour by avoiding the obstacles and keeping a set distance from the wall and gave similar results to that of the software SNN controller.

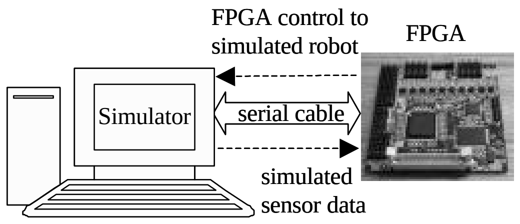

Figure 3. Simulator to FPGA hardware interface.

The FPGA was interfaced to the real robots once the wall following behaviour had been verified using the simulation tools. The robots have a serial port interface and therefore the FPGA code did not have to be changed from that used with the simulator. Figure 4 shows the robot being controlled by the FPGA that is located within the white box towards the top of the left of the robot. Power was supplied to the FPGA directly from the auxiliary supply available on the robot. As expected the tangible robot exhibited similar wall following characteristics as those shown on the simulated robot proving the developmental approach taken to be valid.

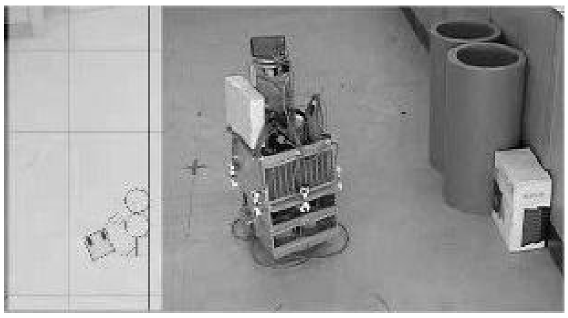

Figure 4. FPGA controlling the tangible robot with corresponding simulator view shown on the left hand side.

6. Future Work and Conclusions

This paper has overviewed the initial research towards building tangible collaborative autonomous agents for fluidic environments. The agent's intelligence was based on spiking neural networks implemented on FPGA devices. The paper has demonstrated a number of key aspects of the project. Firstly FPGAs can be used to implement spiking neural networks to perform navigational tasks. Secondly the spike response can be easily be accurately modeled by using look-up-tables. Thirdly it has been demonstrated that the FPGA can be used in conjunction with simulation tools to control a virtual simulated robot creating a spiking neural network test and development environment. The FPGA linked to the simulation tool could also serve as an accelerator and enable hybrid simulations of software and hardware controlled virtual robots to be studied. Finally it was shown that the FPGA-SNN could be used to control a tangible robot for wall following, yielding similar results to those seen on the simulator. This proved the feasibility of the development and simulation environment that is currently under construction.

Future work will consider the implementation of more complex spiking neural networks and optimization of the neurons onto the NMRC's new miniature 25mm FPGA module [11]. These networks will aim to complete more challenging tasks of fault detection and collaboration in fluidic environments.

A development environment for the graphical input and formal specification of spiking neural networks is under construction and eventually will support automatic translation to VHDL format. The simulation tool will be linked to the development environment and this will support 3D fluidic simulation.

7. References

[1] W. Mass, "Networks of spiking neurons: The third generation of Neural Network Models", *Neural Networks*, Vol. 10(9), 1997, pp 1659-1671.

[2] Ponca M and G. Scarbata, "Implementation of Artificial Neurons Using Programmable Hardware", *Synopsys User Group Conference - SNUG Europe 2001*, 12-13. March, Munich, Germany, 2001.

[3] D. Roggen, S. Hofmann, Y. Thoma, D. Floreano, "Hardware spiking neural network with run-time reconfigurable connectivity", *2003 NASA/DoD Conference on Evolvable Hardware*, 2003, pp. 189-198.

[4] F.J. Pelayo, E. Ros, X. Arreguit, A. Prieto, "VLSI Implementation of a Neural Model Using Spikes", Analog Integrated Circuits and Signal Processing", *Special Issue on Neuromorphic Engineering*, Kluwer Academic Publishers, Vol. 13, 1997, pp. 111-121.

[5] D.H. Goldberg, G. Cauwenberghs, A.G. Andreou, "Analog VLSI Spiking Neural Network with Address Domain Probabilistic Synapses", *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS'2001)*, Sydney, Australia, May 6-9, 2001.

[6] T. Schoenauer, N. Mehrtash, H. Klar, "Architecture of a Neuroprocessor Chip for Pulse Coded Neural Networks", *International Conference on Computational Intelligence and Neuroscience (ICCIN'98), (JCIS'98)*, Research Triangle Park, NC (USA), 1998, pp. 17-20.

[7] T. Lehmann and R. Woodburn, "Biologically-inspired on-chip learning in pulsed neural networks", *Analog Integrated Circuits and signal Processing*, 1999, 18, pp. 117-131.

[8] Wind River webpage: http://www.windriver.com/

[9] M. Colley, G. de Souza, H. Hagras, A. Pounds-Cornish, G. Clarke, V. Callaghan, "Towards developing micro-scale robots for inaccessible fluidic environments", *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 2004.

[10] Xilinx, *Spartan-IIE 1.8V FPGA Family: Complete Data Sheet DS077, Version 2.1*, 2003.

[11] The 25mm cube module, NMRC web page: http://www.nmrc.ie/research/mai-group/25cube_mai.html