RECONFIGURABLE PLATFORMS AND THE CHALLENGES FOR LARGE-SCALE IMPLEMENTATIONS OF SPIKING NEURAL NETWORKS

Jim Harkin[1], Fearghal Morgan[2], Steve Hall[3], Piotr Dudek[4], Thomas Dowrick[3], Liam McDaid[1]

[1]Intelligent Systems Research Centre
University of Ulster, N. Ireland
Email: jg.harkin@ulster.ac.uk

[4]School of Electrical and Electronic Eng., University of Manchester, UK

[3]Institute for Nanoscale Science, Eng. & Tech., University of Liverpool, UK

[2]Bio-Inspired Electronics and Reconfigurable Computing Group, NUI Galway, Ireland

ABSTRACT

FPGA devices have witnessed popularity in their use for the rapid prototyping of biological Spiking Neural Network (SNNs) applications, as they offer the key requirement of reconfigurability. However, FPGAs do not efficiently realise the biological neuron/synaptic models. Also their routing structures cannot accommodate the high levels of neuron inter-connectivity inherent in complex SNNs. This paper highlights and discusses the current challenges of implementing large scale SNNs on reconfigurable FPGAs. The paper presents a novel Field Programmable Neural Network (FPNN) architecture incorporating low power analogue synapse and a network on chip architecture for SNN routing and configuration. Initial results are presented.

1. INTRODUCTION

The basic processing units in the brain are neurons that are interconnected in a complex pattern via synapses [1]. Real biological neurons communicate through pulses or spikes and use the timing of the pulses to transmit information and perform computations [2]. Inspired by biology, researchers aim to implement reconfigurable and highly interconnected arrays of hardware Neural Network (NN) elements to produce powerful and accelerated signal processing units. However, standard topologies employed to model biological SNNs are proving difficult to accelerate in hardware, even for moderately complex networks.

FPGA devices have witnessed popularity in their use for rapid SNNs prototyping [3-6] since they offer the key requirement of hardware reconfigurability. Efficient low area and power implementations of synaptic junctions and neuron interconnect are key to efficient SNN hardware implementations. Current FPGAs cannot accommodate the high levels of inter-neuron connectivity inherent in complex SNNs as their Manhattan-style mesh routing schemes exhibit switching requirements which grow non-linearly with mesh sizes [7]. Network-on-chip (NoC) approaches offer an alternative method to providing inter-neuron connectivity [8]. NoCs employ concepts from traditional computer networking to realise a similar communicating structure on hardware. The key benefit from NoCs is scalable connectivity; higher levels of connectivity can be provided between network nodes without incurring a large interconnect to device area ratio. In this paper the authors present a Field Programmable Neural Network (FPNN) architecture for the realisation of SNNs. The FPNN is centred on a NoC-based neural tile architecture and programmable neuron cell which address the interconnect and bio-computational resources challenges. It will be shown how the proposed FPNN supports the routing, biological computation and configuration of SNN topologies on hardware, and its potential in realising large scale SNNs with a synaptic density significantly in excess of what is currently achievable in hardware [5-6, 9]. Section 2 of the paper provides a review of related work and identifies the challenges of realising large scale SNN hardware implementations. Section 3 introduces the proposed FPNN and section 4 discusses the neural tile. Section 5 presents initial results on the FPNN and neural tile implementation while section 6 provides a summary of the paper and discusses future work.

2. BIO-INSPIRED PLATFORMS & CHALLENGES

SNNs implemented in hardware can take full advantage of their inherent parallelism and offer the potential to meet the demands of real-time applications. FPGAs have previously been used to accelerate biologically inspired computations [3, 5-6, 9], however they are not appropriately suited for SNN implementations as they attempt to map biological neuron and synaptic computations onto general arrays of configurable logic blocks which are not optimised in area-size or power consumption for dense network realisations [6]. Hence, existing FPGAs can only provide limited synapse density. Providing configurable, dedicated computational SNN blocks within an FPGA structure could offer a more suitable reconfigurable platform for SNN implementation, with optimised utilisation of hardware area and power. The extension of such computational blocks to support re-programmability and efficient storage of synaptic weights would address the current demand for devices which enable the accelerated prototyping of large-scale SNNs.

978-1-4244-1961-6/08/$25.00 ©2008 IEEE.

2.1. Scalable Interconnectivity

Increasing SNN neuron density results in a non-linear interconnect growth. For example a 2-layered feed forward fully interconnected network with m neurons per layer exhibits an interconnect density of m^2, which rapidly increases as the number of neurons per layer increases. Neuron interconnection in current FPGAs is typically achieved using diagonal, segmented or hierarchical 2-dimensional routing structures [10], where the switching requirements grow non-linearly with the number of logic units on the device. This FPGA interconnect challenge is a significant limiting factor in the suitability of FPGAs for SNN implementation as devices cannot accommodate the high levels of inter-neuron connectivity. Researchers have investigated several routing optimisations and topologies in attempts to improve the FPGA routing latency and performance [10-11]. Recently, researchers have investigated the use of networking concepts to address the connectivity problem of System-on-Chip using NoC topologies [12], and router architectures (asynchronous [13], circuit/packet and wormhole switching [14]). The key benefit however from using the NoC is scalable connectivity; higher levels of connectivity can be achieved without excessive interconnect-to-device area.

The SNN interconnect problem is similar to that of SoCs; SNNs have large numbers of neurons (typically in excess 1,000 for complex applications), which exhibit high inter-neuron connectivity requirements. Current attempts to realise SNN using time-multiplexed multi-processors in FPGAs have had limited success with only small numbers of neurons implemented due to connectivity issues [4, 6-7]. Multi-processor approaches using network-type communication structures are also under investigation [13]. However, in general these approaches do not accommodate the dedicated computational requirements of the biological neurons and the temporal dynamics of SNN. Reported implementations provide sub-optimal strategies which inefficiently attempt to 'force-fit' SNN into current regular data-path multi-processor architectures.

Several full-custom architectures have been proposed [9, 15] which aim to address the inefficiencies of FPGAs by including optimised synaptic cells and bus-based address-event representation routing structures. However, these architectures cannot scale due to limited connectivity between neurons, and their 'fixed' full-custom nature prohibits reconfigurable SNN topologies.

2.2. Key Challenges

A major challenge in the development of bio-inspired platforms is efficient hardware implementation strategies that can support large scale low-power realisations and re-programmable interconnect. In particular, the problem of SNN inter-neuron connectivity is prohibiting the implementation of biological scale NNs [1]. To address the large scale implementation issues of programmability, weight storage, power consumption, scalability and inter-neuron connectivity requires the creation of a new device architecture tailored to the requirements of SNNs. This paper proposed a custom Field Programmable Neural Network (FPNN) architecture which merges the programmability features of FPGAs and the scalable interconnectivity of NoCs with low-area/power spiking neuron cells. In particular, the FPNN will support the programmability of SNN topologies on hardware providing an architecture which will enable the accelerated prototyping and hardware-in-loop training. Earlier investigations by the authors in using NoC router strategies to realise artificial neural networks [16] has demonstrated benefits in network scalability. In addition, the authors have developed an initial custom low-area/power programmable synapse cell which demonstrates characteristics similar to real biological synapses [17-18].

3. FPNN ARCHITECTURE

The proposed FPNN architecture is illustrated in Fig. 1(a) as a 2-dimensional array of interconnected neural tiles surrounded by I/O blocks. The neural tiles are connected in North, East, South and West directions forming a nearest neighbour connect scheme using a network of NoC routers. Each neural tile can be programmed to realise neuron-level functions. A SNN is realised on the FPNN architecture by programming the tile functionality and connectivity. For example, consider the interconnectivity requirements of a feed-forward (FF) 2-layered nxm SNN network with each neuron in layer 1 connected to m neurons in layer 2. When a neuron in layer 1 fires (spike), its pulse signal is propagated to the target neurons in layer 2 via dedicated individual lines. Using a NoC strategy, the same pattern of connectivity between layers is achieved in the FPNN through time-multiplexing of the communication channels between routers; allowing data to be propagated from source to destination which significantly reducing interconnect density. For example, spike events occurring in layer 1 are forwarded to the associated synapses of layer 2 via several router transmissions. The proposed NoC strategy uses individual routers to group n synapses using a novel structure referred to as a *neural tile*, illustrated in Fig. 1(b), and can be viewed as a macro block of the FPNN. For example, a feed-forward network with n=m=10^3 neurons per layer would require m (10^3) neural tiles each containing n (10^3) synapses; where each tile contains 1 of the post-synaptic neurons of layer 2. The aim of this strategy is to use the router of each tile to communicate spike events to and from its group of n synapses. This enables a reduced number of connections; for example, an SNN interconnect density of 10^6 (n x m) can be implemented using 4x10^3 (4 x *m connections*).

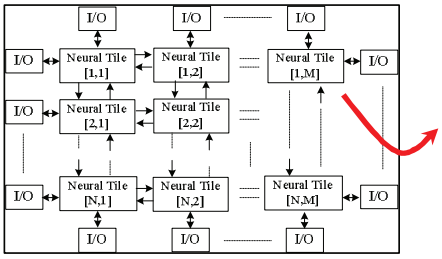

Fig.1 (a) FPNN structure

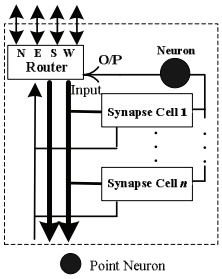

Fig.1 (b) Neural tile

The *n* individual synapses in a neural tile are referred to as *synapse cells* and are combined with the point neuron to form the neuron cell. The synapse cell is analogue in nature and captures the pertinent features of real biological synapses [17]. The inputs and outputs to the synapse cells of a tile are controlled via the NoC router, whereby the spike-events of a spike-train are received/transmitted in the form of data packets from the neighbouring routers of a tile. Exploiting the relatively low-frequency of biological spike trains (~Hz) [2] enables the time-multiplexing of spike data on router paths between layers; enabling large parallel networks to be realised on the regular FPNN structure without physically incurring the traditional connectivity requirements. The proposed method of connecting tiles also enables multi-layered feed-forward and recurrent networks SNN topologies to be realised.

4. FPNN NEURAL TILE

The FPNN neural tile is illustrated in Fig.2 and highlights the connections between the NoC router, synapse cells and neuron. The packet-switched router implements 12-bit communication paths with buffer support. A round-robin scheduling policy is currently used although multicasting routing is currently being investigated to support spike-event traffic. There are 6 main communication buses/lines:

- *Spike I/P*: initiates a spike on individual synapse cells.
- *Spike O/P*: receives spike events from the neuron.
- *Mode*: specifies runtime or configuration mode.
- *Indexing*: used to address individual synapse cells for receiving spike events or configuration data.
- *ACK*: acknowledges the correct synapse addressing.
- *Config Data*: feeds configuration data to the cells.

Each tile router has a unique address and the synaptic connectivity is specified using an Address Table (AT). The AT is programmed to specify the desired connectivity (topology) between tiles enabling spike events to be routed; a spike event is detected at the *Spike O/P* and the AT identifies which tiles must receive event notification. This data (source\target neuron) is transmitted in packets via tile routers until it is received by its destination tiles. The neural tiles operate in one of two modes: *runtime* or *programming*. In runtime mode the FPNN routes spike events and computes the programmed SNN functionality. In programming mode the FPNN is configured to realise particular neuron/synapse models and the desired SNN topology. Configuration data is delivered to the FPNN device in the form of data packets and contains information on the configuration of the router's AT, and the selection of cell synapse weights via programmable voltage lines V_q.

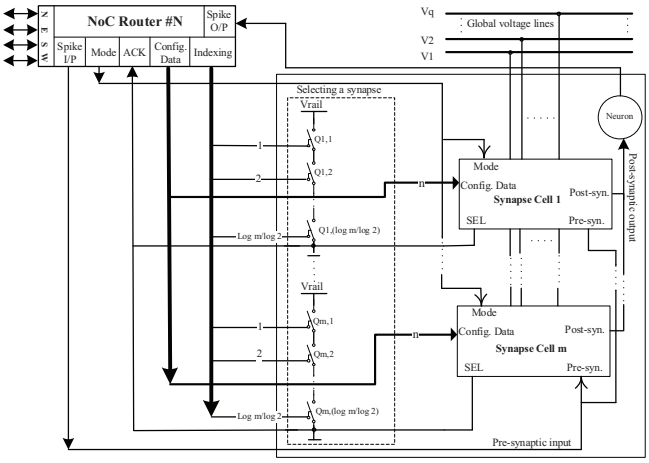

Fig.2 Neural tile

A novel weight distribution and configuration storage architecture is presented in Fig.3 which allows any number or combination of the P synapses to be permanently hardwired to any weight voltage level, V_q. Each synapse captures key pertinent features of real synapses [17] such as long and short term plasticity, whereby for the latter, the synapse can be programmed to operate in either the facilitation or depression states. In keeping with biological plausibility, weight updates for long term plasticity should be governed by a Hebbian-based rule [18], whereby each synapse can be configured with a weight voltage V_q, whose value is derived from this rule after a period of training.

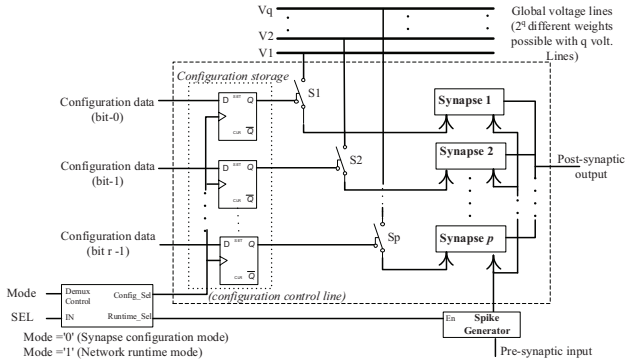

Fig.3 Synapse cell architecture

Currently the authors envisage an off-line training procedure whereby the synapse weights are determined by an appropriate algorithm and then the switches, S_1-S_p, are programmed via storage latches to hard-wire a weight to an associated synapse. This enables the accumulation of *x*

(where $1 \leq x \leq p$) synaptic currents to provide an overall weight.

5. PERFORMANCE

Programmable synapse and neuron test circuits are currently being fabricated by Europractice. For an indicative tile with 10 programmable synapse cells and one neuron, a compact area size of 3 x 11µm using 90nm CMOS technology can be achieved (configuration storage, voltage lines and NoC router not included). The small area exhibited by the synapse offers the potential to realise large scale networks as its area overhead is not significant in comparison to the router. Moreover, the synapse power consumption is small. Earlier work by the authors [18] estimates that for a network containing 10^6 synapses synchronised to an input spike train frequency of 1MHz will yield a power consumption/sec in the of order of 10^{-4} Watts. Although the estimation does not include the power consumed by configuration circuitry, it does illustrate that the FPNN is very energy efficient.

An initial router design has been implemented on a Xilinx XC2V1000 [16]. Data from [19] estimates the AEthereal NoC router implementation of 2,658 LUTs on the Virtex-4 to be equivalent to 2.28mm^2 using 90nm CMOS technology. Using this data the authors can conservatively estimate the router of 234 LUTs (synthesized for Virtex-4) to be a factor of 11.36 smaller in area than the AEthereal and therefore, equal to 0.201mm^2. This analysis provides an early area estimate of the proposed neural tile at 201.033×10^{-3} mm^2 $\{33 \times 10^{-6} + 201 \times 10^{-3}\}$. This initial estimate suggests promising results for FPNN networks, since the tile area is relatively small and supports a regular layout for interconnectivity. Research is currently under way to optimise the tile area further: for example, Fig.4 shows that by increasing the numbers of neurons in a tile reduces the number of routers required for SNN implementations. This reduces the overall area size of the FPNN.

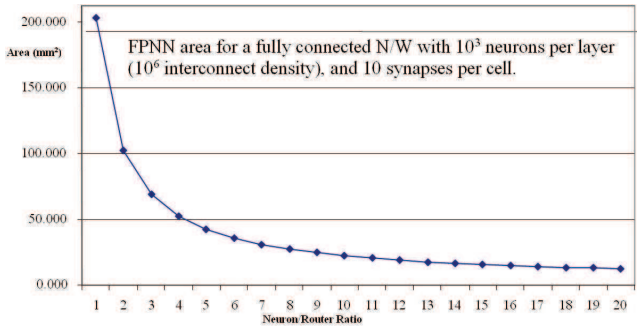

Fig.4 Scalability with increased ratio of neurons per tile

Note that the above calculation does under estimate the total FPNN area but merely highlights a scaling trend for the FPNN tile.

6. CONCLUSION AND FUTURE WORK

The challenges for large scale hardware SNN implementations have been highlighted. A novel FPNN hardware architecture has been presented utilising NoC routers with a low power/geometry neuron cell to provide programmable/scalable interconnect and resources. Initial results demonstrated the scalability of the FPNN in terms of area and power. Future work will explore NoC routing polices and topologies for spike-traffic. Approaches to training will be investigated, and the integration of the NoC router and neural tile configuration architecture with the neuron cell will be realised.

7. REFERENCES

[1] A Hodgkin et al; *Journal of Physiol*., 117, pp. 500-544, 1952

[2] W Maass, "Computation with Spiking Neurons: The Handbook of Brain Theory and NNs", 2nd Ed, MIT Press, 2001

[3] A Ghani A, J Harkin et al.; "Area Efficient Architecture for Large Scale Implementation of Biologically Plausible Spiking Neural Networks on Reconfigurable Hardware", *FPL*, 2006

[4] B Glackin et al., "Novel Approach for the Implementation of Large-Scale SNNs on FPGAs", *Artificial NNs Conference*, 2005

[5] A Upegui et. al; "An FPGA platform for on-line topology exploration of SNNs", *Micros & Microsystems*, 29(5), 2005

[6] MJ Pearson et al., "Implementing SNNs for Real-Time Signal-Processing and Control Applications: A Model-Validated FPGA Approach", *IEEE Trans. on Neural Nets*, 18(5), 2007

[7] LP Maguire, J Harkin et al.; "Challenges for large-scale implementations of SNNs on FPGAs", *Neurocomputing*, 71(1-3), pp. 13-29, 2007

[8] L Benini, G DeMicheli, "Networks on Chips: A New SoC Paradigm", *IEEE Computers*, pp. 70-78, Jan 2002

[9] RJ Vogelstein et al., "Dynamically Reconfigurable Silicon Array of Spiking Neurons with Conductance-Based Synapses", *IEEE Trans. Neural Nets*, 18 (1), pp. 253 – 265, 2007

[10] A DeHon, R Rubin, "Design of FPGA Interconnect for Multilevel Metallization", *IEEE Tran. VLSI Sys*., 12(10), 2004

[11] J Rose, et al. "Using Bus-Based Connections to Improve FPGA Density for Implementing Data-path Circuits", *IEEE Transactions VLSI Systems*, 14(5), pp. 462-473, 2006

[12] S Jovanovic et al. "CuNoC: A Scalable Dynamic NoC for Dynamically Reconfigurable FPGAs", *FPL*, 2007, pp. 753 - 756

[13] AD Rast et al, "Virtual Synaptic Interconnect Using an Asynchronous Network-on-Chip", *WCCI*, June 1st-6th, 2008

[14] C Schuck et al., "artNoC - A Novel Multi-Functional Router Architecture for Organic Computing", *FPL*, 2007, pp. 371–376

[15] J Schemmel et. al, "Mixed-Mode Analog NN Using Current-Steering Synapses", *Analog Circ. & Signal Proc.*, Vol. 38, 2004

[16] J Harkin et al. "Novel Interconnect Strategy for Large Scale Implementations of NNs", *IEEE Soft Comp. in Indust. App*, 2007

[17] Y Chen, LJ McDaid et al, "A Programmable Facilitating Synapse Device", *WCCI*, June 1st-6th, 2008

[18] Y Chen, et al. "A Silicon Synapse Based on A Charge Transfer Device for SNN", *Int. Symp. on Neural Nets*, May 2006

[19] K Goossens, et al., "Hardwired NoCs in FPGAs to unify Data & Config. Interconnects", *Int Symp. on NoCs*, April 2008