Exploring Spiking Neural Network on Coarse-Grain Reconfigurable Architectures

Hassan Anwar
Polytechnique Montreal
hassan.anwar@polymtl.ca

Syed M. A. H. Jafri
University of Turku, Finland
Royal Institute of Technology
jafri@kth.se

Sergei Dytckov
University of Turku
Sergei.dytckov.utu.fi

Masoud Daneshtalab
University of Turku, Finland
Royal Institute of Technology
masdan@utu.fi

Masoumeh Ebrahimi
University of Turku, Finland
Royal Institute of Technology
mebr@kth.se

Ahmed Hemani
Royal Institute of Technology
hemani@kth.se

ABSTRACT

Today, reconfigurable architectures are becoming increasingly popular as the candidate platforms for neural networks. Existing works, that map neural networks on reconfigurable architectures, only address either FPGAs or Networks-on-chip, without any reference to the Coarse-Grain Reconfigurable Architectures (CGRAs). In this paper we investigate the overheads imposed by implementing spiking neural networks on a Coarse Grained Reconfigurable Architecture (CGRAs). Experimental results (using point to point connectivity) reveal that up to 1000 neurons can be connected, with an average response time of 4.4 msec.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Connectionism and neural nets; B.1.4 [**Hardware**]: Microcode Applicaitions—*Special-purpose*; B.1.4 [**Hardware**]: Interconnections—*Topology*

1. INTRODUCTION

The capability of learning has been always an important topic in different areas [3]. Among them, neural networks becoming an increasingly popular technique to realize learning robotics. Use of neural networks in robotics requires a fast and efficient implementation platform. Recently, the increasing speed and high performance requirements, coupled with the demands for flexibility and low non-recurring engineering costs, have made reconfigurable hardware a very popular implementation technology for neural networks [4]. Today reconfigurable architectures enable partial and dynamic runtime self-reconfiguration [12]. This feature allows the substitution of neural network on a hardware design implemented on this reconfigurable hardware, and therefore, a single device can be adapted to implement various functionalities by simply uploading a new configuration based on neural network application. The reconfigurable architectures can be classified depending on their granularity, i.e., the number of bits which can be explicitly manipulated by the programmer. Coarse-Grained Reconfigurable Architectures (CGRAs) provide operator level configurable functional blocks, word level datapaths, and powerful and very area-efficient datapath routing switches. Compared to fine-grained architectures (like FPGAs) CGRAs enjoy massive reduction of configuration memory and configuration time (two or more orders of magnitude) as well as considerable reduction in routing and placement allocation. All this also results in a significant reduction of the overall area (from 66% to 99.06%) and energy consumed per computation (from 88% to 98% [10]), though at the cost of a loss in flexibility compared to bit-level operations. Therefore, CGRAs have been a subject of intensive research since the last decade.

Most of the existing works that attempt to speed up the neural network algorithm employ FPGAs [9]. They speed up computations by exploiting the parallel processing offered by the FPGAs. However, the FPGAs require fine-grained connectivity that is inefficient to handle the complex interconnections required by the neural networks [5]. Compared to FPGAs, Networks-on-Chip (NoCs) simplify the connectivity [5], thereby enhancing scalability. However, the NoC-based solutions (using processor with every node) are inefficient and require complex switches. Therefore, while NoC-based solutions provide high scalability, they are unable to provide the high speed parallel processing offered by FPGA. In this paper we explore the feasibility of implementing the neural networks on a CGRA. The proposed solution combines the parallel processing of FPGA-based implementations with that of the scalability similar to generic NoCs. Specifically, we consider a phenomenon known as Spike-Timing-Dependent Plasticity (STDP). To evaluate experimentally the efficiency of the proposed solution on a CGRA, we have chosen the Dynamically Reconfigurable Resource Array (DRRA) [10]. Nevertheless, the results obtained in this paper should essentially be applicable to most grid based CGRAs as well.

This paper has two major contributions:

1. We present the architecture and implementation of the spiking neural networks on a coarse grained reconfigurable array (CGRA).
2. We analyze the scalability and overhead of the proposed technique, on an actual CGRA called Dynamically Reconfigurable Resource Array (DRRA).

2. RELATED WORK

Implementation of SNN on FPGA is explored in [6], showing that even the highly optimized digital emulation of neuron consumes significant area on FPGA. With 1 to 100 connectivity, only 18 neurons could be implemented in parallel. To utilize larger network sizes time-multiplexing schemes can be used. The MicroBlaze soft processor with dedicated synaptic and neuron IPs was proposed to manage time-multiplexing and off-chip memory access. Instantiation of multiple processors on two parallel FPGA chips was capable of emulating 268,656 neurons in real time. SIMD array of neural processing elements connected through a bus was utilized in [7]. With the maximum connection rate of 1 to 16, 7000 of LIF neurons with inter-neural propagation delays and noisy synaptic integration could be emulated at real time. However, the maximum spiking rate was limited to 50 Hz. Experiments with 32 neurons connected through an AER bus with the included STDP learning showed the speedup of 3125 over the real time in [2]. By moving neural variables to off-chip memory and applying time-multiplexing on that architecture, the implementation of a million of neurons in real time was established, but learning possibility was excluded [1].

3. PRELIMINARIES

In this paper we will attempt to implement a phenomena known as Spike-Timing-Dependent Plasticity (STDP) using Spiking Neural Networks (SNN). In order to verify our results, we used the CGRA platform named Dynamically Reconfigurable Resource Array (DRRA). The computational systems based on SNN is getting critical, demanding improved and faster computations. The CGRA becomes a promising solution for providing a scalable and robust interconnection fabric.

3.1 Spiking Neural Network (SNN)

Spiking Neural Networks (SNN) are the latest generation of neural networks that emulate the behavior of biological neurons. For this work, we have chosen the simplest and the most widely used SNN model, called Leaky Integrate and Fire, given by Equation $dv/dt = I + a - bV$. Where the potential V integrates input spikes I and leaks over time with $-bV$ component. Coefficient a determines equilibrium point and coefficient b the speed of leakage. When the threshold potential reaches a neuron output, a spike and its potential is reset. The neurons are connected in one-to-many fashion with weighted connections (called synapses). Spikes carry the events while synapses are responsible for learning and generating weighted inputs. Learning is performed by adjusting synaptic weights. In this paper we have modeled Hebb's learining technique called Spike-Timing-Dependent Plasticity (STDP), shown in Equation 1 [11]. This rule increases a synaptic weight if the time difference ($\triangle t$) between an output (post- synaptic) and an input (pre- synaptic) spike is positive (i.e. when a pre- synaptic spike arrives before a post- synaptic one) and vice versa.

$$\begin{cases} A_{+}exp(- \triangle t/\tau_{+}), & \text{if } \triangle t > 0. \\ A_{-}exp(- \triangle t/\tau_{-}), & \text{if } \triangle t < 0. \end{cases} \tag{1}$$

3.2 Dynamically Reconfigurable Resource Array (DRRA)

DRRA computational layer is shown in Fig. 1. It is composed of four elements: (i) Register Files (reg-files), (ii) morphable Data Path Units (DPUs), (iii) circuit-Switched Boxes (SBs), and (iv) sequencers. The reg-files store data for DPUs. The DPUs are functional units, responsible for performing computations. SBs provide interconnectivity between different components of DRRA. The sequencers hold the configware, which corresponds to the configuration of the reg-files, DPUs, and SBs. Each sequencer can store up to 64 36-bit instructions and can reconfigure the elements only in its own cell. As shown in Fig. 1, a *cell* consists of a Reg-file, a DPU, SBs, and a sequencer, all having the same row and column number as a given cell. The configware loaded in the sequencers contains a sequence of instructions (reg-file, DPU, and SB instructions) that implements the DRRA program.

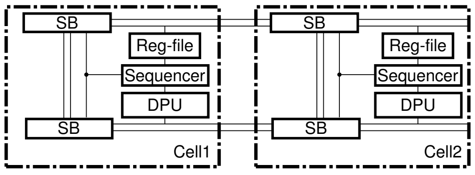

Figure 1: DRRA computation Layer

4. MODELLING SPIKING NEURAL NETWORKS ON DRRA

Table 1 summarizes how various spiking neural network components, described in Section 3.1, are implemented on DRRA platform. To mimic the neuron functionality, we have enhanced the functionality of the Data Path Unit (DPU) and the reg-files. Using the reg-file to receive data from other neurons, the DPU performs the computations on the received data. Since the spiking neural networks require timing information, we have added a counter and a SNN unit (explained later in Section 4.2). To allow communications between neurons, we exploit the circuit network provided by DRRA, unchanged. Finally, since the spiking neural networks require point to point connectivity (while DRRA register files have only two ports), we have used sequencers to implement point to point communication between multiple neurons (the architecture is discussed in detail in Section 4.1).

Table 1: Summary of how various neural network components are realized

Neural network entity	DRRA implementation
Neuron/Synapse	DPU + Regfile
Inter neural communications	Circuit switched interconnect+ sequencer

4.1 Inter Neural Communication

Neural networks require point to point communication between multiple neurons. To implement these connections on DRRA, we have to consider two architectural properties: (i) every DRRA component has only two read/write ports and (ii) a DRRA component can be directly connected to a component at most three hops away. Since these architectural characteristics were designed after a careful evaluation of area/power trade-off, we decided not to modify them. The point to point connectivity was realized by using time division multiplexing. In the proposed approach a specific time slot was assigned to each pair of neurons. To allow a scalable solution, we have chosen a hierarchical clustered approach shown in Fig. 2.

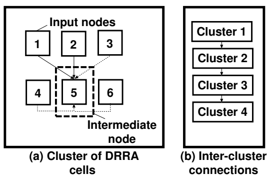

Figure 2: Inter neural communication realization

Fig. 2 (a) shows a cluster of 6 DRRA cells. In the cluster, one of the cells is chosen as an intermediate node. The intermediate node receives data from the 6 cells in the cluster (including itself). To allow connections with 6 cells on a 2-port component, we exploit time division multiplexing combined with partial and dynamic reconfiguration. In the overall process the intermediate cell receives data from 2 cells at a time and then shifts to the other cells. The process continues till the data from all the cells is received. Once every cluster has received inputs, the intermediate nodes communicate with each other serially to ensure point to point connectivity. Fig. 3 shows the instructions in DRRA sequencers (i.e. needed to implement the time division multiplexing). The figure shows that 5 cycles are needed to collect information from four DRRA cells in the cluster. It should be noted that 2 additional cycles are required to reconfigure the circuit switched network. The inter cluster communication takes two cycles for reconfiguration and an additional cycle to transmit data.

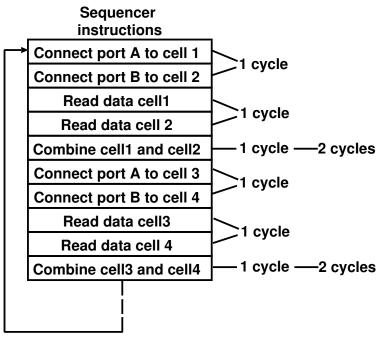

Figure 3: Sequencer instructions to implement time division multiplexing for four DRRA cells

4.2 Neuron/Synapses Realization

4.2.1 *Processing requirements formalization*

When a signal is received (or sent) by a neuron, additional processing is required. The processing requirements are dependent on whether a signal is received (pre-synapse spike) or transmitted (post- synapse spike). In this section, we will formulate the processing requirements of each spike separately. To formulate processing requirements of pre-synapse spike, let S_{pre} be the received pre-synapse spike received by neuron N_i from a neuron N_{i-1}. Upon occurrence of S_{pre}, four operations are performed sequentially: (i) the spike arrival time, T_{pre}, is stored in N_{i-1}^{th} index of a dedicated pre-synapse array, $A[N]$, where N is the number of neurons, (ii) the time when last post-synaptic spike was transmitted, T_{post}, is retrieved, (iii) the difference between the spike times, $\triangle T = T_{pre} - T_{post}$, is calculated, and (iv) the result is stored in a weights array $W[N]$. Where τ determines the time intervals when changes of a weight occur and A is constant that determines the maximal change. When the post-synaptic spike, S_{post}, is transmitted, three sequential steps follow: (i) the spike transmitting time, T_{post}, is stored N_{i-1}^{th} index of a dedicated register. (ii) the difference between the spike times, $\triangle T = A[i] - B$, is calculated for each entry in $A[N]$, and (iii) the result calculated by Equation 1 is stored in a weights array $W[N]$.

4.2.2 *Pre/post-synapse spike realization*

Figures 4, 5, and 6 show how the spiking neural network was realized on a DRRA platform. The data path needed to process the pre/post-synaptic spikes is shown in Figures 4 and 5. We have used the reg-files to mimic $A[N]$. Since each reg-file is composed of 64 registers, to realize a system greater than 64 neurons, we have used DiMArch memory (Explained in [8]). Therefore, for connecting more than 64 neurons, an additional overhead of 8 cycles is incurred to send and receive data from the DiMArch. Implementing SNN, requires a divider, that was missing in the existing DRRA implementation. To realize a divider, we considered two alternatives: (i) to implement the divider using shift registers at the cost accuracy or (ii) to implement a divider in hardware. Since for the real-time robotics (targeted in this paper) speed and power (per computation) are far more significant than silicon efficiency, we opted for a dedicated double precision floating point divider (shown in STDP block Figure 4) to allow quick convergence at low power. In addition, we implemented counters that allows to transmit the spikes in terms of the time stamps. Finally, the state machine that realizes the sequential steps of pre/post synaptic spikes is implemented.

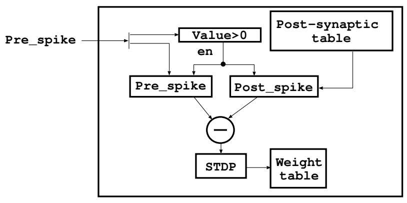

Figure 4: Dedicated hardware for pre-synaptic spike analysis

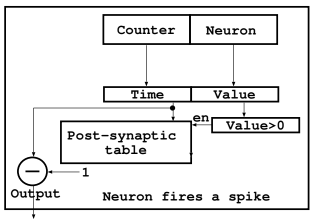

Figure 5: Dedicated hardware for post-synaptic spike analysis

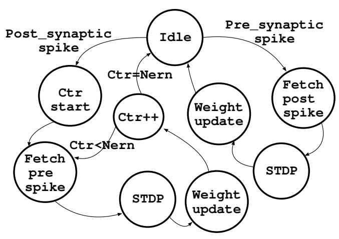

Figure 6: State machine for pre/post-synaptic spike analysis

5. EXPERIMENTAL RESULTS

To investigate the efficacy of the proposed approach, we implemented SNNs on the enhanced DRRA platform. Specifically we evaluated the scalability of the proposed approach and its overheads.

5.1 Scalability analysis

To determine the scalability of implementing massively parallel neural networks, we measured the cycles needed to realize point to point connectivity using time division multiplexing. We mapped the solution using different Cluster Sizes (CS). A cluster is defined as the number of neurons that can communicate directly with each other in parallel (using space division multiplexing). Since DRRA only permits up to 3 hop connectivity, the maximum cluster size was limited to 5 cells. The scalability was analyzed by calculating the latency for different number of neurons as shown in Fig. 7. As expected, a completely serial implementation (cluster size=1) for 200 neurons required maximum latency of 336,000 clock cycles compared to latency with cluster size=5 (i.e. 67,200 for 200 neurons). Considering that DRRA operates at a frequency of 400 MHz, the connectivity for 200 neurons can be achieved in 1.68 msec. Extrapolating the results a connectivity of up to 1000 neurons can be achieved in 4.4 msec. To evaluate additional cells needed for parallel implementation, we analyzed the additional intermediate cells needed for different cluster sizes as shown in Fig. 8. From these figures, it can be concluded that the minimum area consumption is observed for cluster size=1 running 20 parallel neurons and maximum area consumption is observed at cluster size=5 running 200 neurons.

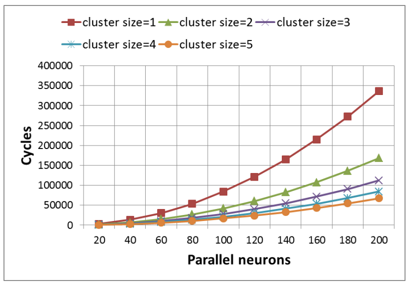

Figure 7: Parallel neurons computational latency

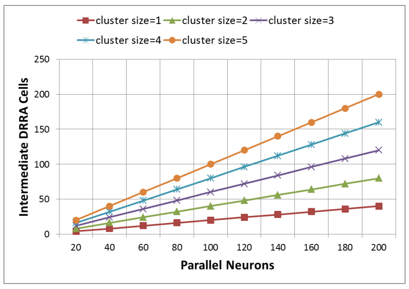

Figure 8: Cells consumption

6. OVERHEAD ANALYSIS

To estimate the area and power overhead of implementing SNNs, we synthesized the DRRA fabric with enhanced hardware support for 65 nm technology at 400 MHz frequency using Synopsys Design Compiler. Fig. 9 illustrates the total area and power (pow) usage of the design. The maximum area utilization is for DRRA cells and the area overhead of the system is about 39 % (divider, datapath, and state machine). Similarly, the maximum power consumption is achieved by the DRRA cells and the power consumption for divider, datapath, and state machine is 27 % of the overall design.

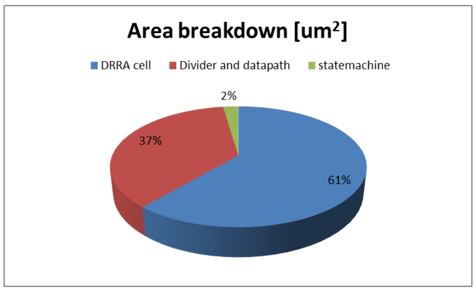

Figure 9: Area consumption of the design.

7. CONCLUSION

In this paper, we presented the architecture and implementation of neural networks on a CGRA. The motivation for choosing CGRA as an implementation platform is that it has potential to dominate its counterpart FPGA in near future (for applications needing high performance). The neural networks were realized by connecting dedicated hardware blocks (to mimic the neurons functionality) in a time-multiplexed manner. To tackle complexity, we proposed clustering. Experimental results revealed that 1000 neurons can be connected in one to all fashion, with a net response time of 4.4 msec.

8. ACKNOWLEDGMENTS

This work was supported by Regroupement Strategique en Microsystemes du Quebec (ReSMiQ), Nokia Foundation, and VINNOVA (Swedish Agency for Innovation Systems) within the CUBRIC and ERoT projects.

9. ADDITIONAL AUTHORS

Additional authors: Juha Plosila (University of Turku, Finland, email: `juplos@utu.fi`), Hannu Tenhunen (University of Turku, Finland, email: `hannu@kth.fi`) and Giovanni Beltrame (Polytechnique Montreal, Canada, email: `giovanni.beltrame@polymtl.ca`).

10. REFERENCES

[1] A. Cassidy, A. Andreou, and Georgiou. Design of a one million neuron single fpga neuromorphic system for real-time multimodal scene analysis. In *45th Annual Conference on Information Sciences and Systems (CISS)*, 2011.

[2] A. Cassidy, S. Denham, P. Kanold, and A. Andreou. Fpga based silicon spiking neural array. In *Biomedical Circuits and Systems Conference*, 2007.

[3] M. Ebrahimi and A. Jahangireyan. Aerodynamic optimization of airfoils using adaptive parameterization and genetic algorithm. *Optimization Theory and Applications*, 2013.

[4] P. Garcia, K. Compton, M. Schulte, E. Blem, and W. Fu. An overview of reconfigurable hardware in embedded systems. *EURASIP J. Embedded Syst.*, 2006(1):13–13, Jan. 2006.

[5] J. Harkin, F. Morgan, S. Hall, P. Dudek, T. Dowrick, and L. McDaid. Reconfigurable platforms and the challenges for large-scale implementations of spiking neural networks. In *Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on*, pages 483–486, 2008.

[6] L. Maguire, T. McGinnity, B. Glackin, A. Ghani, A. Belatreche, and J. Harkin. Challenges for large-scale implementations of spiking neural networks on fpgas. *JNEUCOM*, 71:13–29, 2006.

[7] M. Pearson, A. Pipe, B. Mitchinson, K. Gurney, C. Melhuish, I. Gilhespy, and M. Ni-bouche. Implementing spiking neural networks for real-time signal-processing and control applications: A model-validated fpga approach. *IEEE Transactions on Neural Networks,*, 18:1472–1487, 2007.

[8] Removed for blind review. In *Proc. Application Specific Systems Architectures and Processors (ASAP)*, Washington, D.C., USA, 5–7 June 2013.

[9] H. Rostro-Gonzalez, G. Garreau, A. Andreou, J. Georgiou, J. Barron-Zambrano, and C. Torres-Huitzil. An fpga-based approach for parameter estimation in spiking neural networks. In *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, pages 2897–2900, 2012.

[10] M. A. Shami. *Dynamically Reconfigurable Resource Array.* PhD thesis, Royal Institute of Technology (KTH), Stockholm, Sweden, 2012.

[11] S. Song and L. Abbott. Cortical development and remapping through spike timing-dependent plasticity. *Neuron*, 32(2):339 – 350, 2001.

[12] Syed M. A. H. Jafri, K. Paul, A. Hemani, J. Plosila, and H. Tenhunen. Compact generic intermediate representation (CGIR) to enable late binding in coarse grained reconfigurable architectures. In *Proc. International Conference on Field-Programmable Technology (FPT)*, pages 1–6, Dec. 2011.