PERFORMANCE COMPARISON OF A BIOLOGICALLY INSPIRED EDGE DETECTION ALGORITHM ON CPU, GPU AND FPGA

Patrick Dempster, Thomas. M. McGinnity, Brendan Glackin and Qingxiang Wu
Intelligent Systems Research Center, University of Ulster, Northland Road, Derry, U.K.

Keywords: SNN, FPGA, GPU, CUDA, CPU, Multithreaded, SNN, Bio-inspired.

Abstract: Implementation of Spiking neural networks (SNNs) are becoming an important computational platform for bio-inspired engineers and researchers. However, as networks increase in size towards the biological scale. Ever increasing simulation times are becoming a substantial problem. Efforts to simulate this problem have been many and varied. Modern Graphic Processing Units (GPUs) are increasingly being employed as a platform, whose parallel array of streaming multiprocessors (SMs) allow many thousands of lightweight threads to run. This paper presents a GPU implementation of an SNN application which performs edge detection. The approach is then compared with an equivalent implementations on an Intel Xeon CPU and an FPGA system. The GPU approach was found to provide a speed up of 1.37 times over the FPGA version and an increase of 23.49 times when compared with the CPU based software simulation.

1 INTRODUCTION

Neuroscientists and computer engineer strive to develop systems which take as their inspiration the most advanced processing system on the planet, namely the human brain. The human brain is a hugely complex, massively parallel processor which is estimated to have in the region of 10^{11} neurons with an estimated 10^{14} connections between them (Khan et al., 2008). A neuron is the basic processing element of the brain, with different models available such as the conductance based integrate and fire (Maguire et al., 2007), Izhikevich (Izhikevich, 2003), and Hodgkin Huxley (Gerstner and Kistler, 2002). Each of these provide a mathematical model which, to varying degrees of complexity, attempt to account for the complex behaviour which is observed when studying the brains functionality. Spiking neural networks (SNNs) are emerging as a paradigm, which have the potential to create biologically plausible systems (Glackin et al., 2009b). Unlike artificial neuron networks (ANNs), SNNs use timing of the spikes to convey information and perform computations.

Networks of SNNs, tend to be highly parallel systems which when simulated on conventional CPU based systems are executed sequentially. As traditional CPUs process information in a a sequential fashion this can cause simulation times to be significant (Khan et al., 2008). Simulation times will increase as emulated systems move towards biological scale, despite the increasing power of conventional CPUs. An approach currently used to increase the number of neurons and synapses which can be simulated, while decreasing simulation time is that of parallel systems. Leveraging these parallel systems, which include beowolf clusters, super-computers, neuromorphic hardware, FPGAs and GPUs to take advantage of the parallel nature of their computation to accelerate simulation. Recently developed GPUs which include support for parallell programming include IBM's cell processor, Nvidia's Telsa class CUDA capable GPUs and ATI Stream Processors.

This paper will present work which has used an Nvidia based GPU to decrease the time required to implement and evaluate an SNN network designed to perform biologically inspired edge detection on an image. Section 2 Provides an introduction to the Compute Unified Device Architecture (CUDA) which is a current approach to programming NVIDIA graphics processing units (GPUs). Section 3 will provide an overview of the SNN application that was used to test the performance of the CUDA GPU approach. Section 4 will report on the experimental results obtained from the implementation of the SNN on the CUDA hardware, a previously reported FPGA implementation and a CPU implementation. Section 5 outlines some initial conclusions of this work and the direction

Dempster P., M. Mcginnity T., Glackin B. and Wu Q..
PERFORMANCE COMPARISON OF A BIOLOGICALLY INSPIRED EDGE DETECTION ALGORITHM ON CPU, GPU AND FPGA.
DOI: 10.5220/0003078904200424
In *Proceedings of the International Conference on Fuzzy Computation and 2nd International Conference on Neural Computation* (ICNC-2010), pages 420-424
ISBN: 978-989-8425-32-4

of further research which will use GPU based CUDA programming to explore large scale SNNs, using different bio-inspired neuron models.

2 TESLA GPU'S AS A PLATFORM FOR PARALLEL PROCESSING

Graphics Processing Units (or GPUs) are a hardware unit traditionally used by PCs to render graphics information to the user. The information presented by displays has evolved as the power of GPUs has increased, from the early PC systems which provided an almost typewriter class monotone display to immersive 3D displays found in advanced workstation PCs today. This evolution has lead to the design, by Nvidia, of modern graphics cards which are in essence massively multi-threading processors with high memory bandwidth (Kirk and Hwu, 2010). The power of GPU has traditionally been used to improve the appearance of the GUI in computer systems and to improve the visual quality of games. However, as interest with in research communities has evolved to harness the power of parallel systems, GPUs have been seen as a way to bring huge amounts of processing power to an individual desktop. Nvida have responded by developing the Compute Unified Device Architecture (CUDA) which allows programmers to better leverage the parallel processing capability of GPUs. In turn this has allowed GPUs to claim a place within the high performance computing family of systems and also lower the cost of entry from thousands of pounds to hundreds of pounds.

As a result of the evolution of high performance graphics systems, there are available Nvidia GPUs whose characteristics allow for the parallel execution of multiple thousands of lightweight threads. Researchers are currently working on ways to harness the power of these lightweight threads, so that they may be used to simulate many thousands of neurons on a GPU (Nageswaran et al., 2009).

Nvidia GPUs which can be programmed using the CUDA, framework typically contain arrays of Streaming Multiprocessors (SMs), with upto 30 SMs available on the largest Nvidia Telsa devices. This capability is paired with upto 4GB of memory, which results in super computer class performance on desktop PCs. Each of the Streaming Multiprocessors available in a Telsa GPU typically contain, eight floating point Scalar Processors (SPs), a Special function unit (SFU), a multi-threaded instruction unit, 16KB shared memory which can be managed by the user, and 16KB of cache memory. GPU also contain a hardware scheduling unit which selects which group of threads (in Nvidia terminology this is called a 'warp' of threads) to be run on the SM. If a single thread within the warp requires access to data which is held in external memory then the hardware scheduling unit can mark another warp of threads to run on the SM, while the data for the thread in the first group is retrieved from external memory, thus helping to mask the memory access time and improve overall performance.

3 SNN ARCHITECTURE FOR EDGE DETECTION

In order to evaluate the potential for simulation performance improvements which may be obtainable when using GPUs as an acceleration platform, an application in emulating neural networks was required. The application which was selected for comparison was the 'SNN for edge detection application' previously reported by (Wu et al., 2007). This application has been subsequently implemented on a field programmable gate array (FPGA), system which is a powerful parallel hardware environment. When combined with a powerful novel reconfigurable architecture (Glackin et al., 2009a), this architecture showed impressive speed increases. Over an order of magnitude speed increase was recorded increase over a similar CPU based implementation of the SNN edge detection application when running on a Intel Xeon class processor. The SNN architecture of this application will now be briefly described in the rest of this section.

The principle upon which the SNN application was designed is to use receptive fields tuned to up, down, left, right orientations to detect the edges contained within the SNN input image.

As can be seen in figure 1 the edge detection application contains a number of 'layers'. In 'Receptor layer' each node represents the current value obtained when converting the pixel value at the corresponding location in the input image. Each value in this layer is then forwarded on to the intermediate layer 'N', via 5x5 receptive field (RF) weight distributions, such that one excitatory and one inhibitory field is formed for each orientation direction and are labelled as 'Δ' and 'X' respectively. Thus, eight RF orientations were used RF_{up_exc}, RF_{up_inh}, RF_{down_exc}, RF_{down_inh}, RF_{left_inh}, RF_{left_exc}, RF_{right_inh}, RF_{right_exc}. Figure 2 shows the weight distribution matrices for each of the orientation selective receptive fields.

As indicated in figure 3, there are 50 RF connections from each input pixel in the 'Intermediate' (figure 1) layer. Each value in the input layer (x,y) is con-

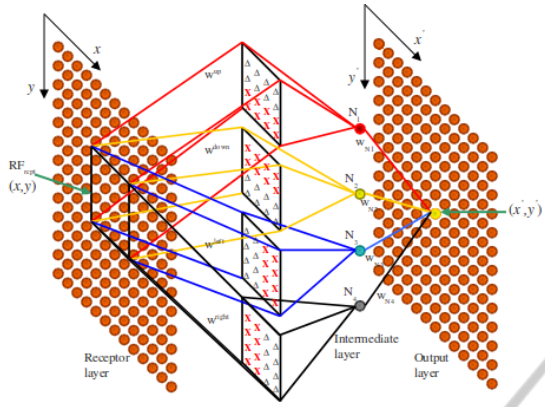

Figure 1: SNN Model for Edge Detection (Wu et al., 2007).

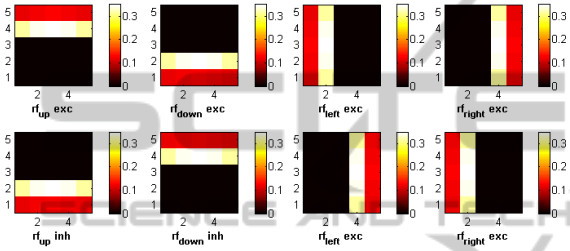

Figure 2: RF edge orientated weight distributions (Glackin et al., 2009a).

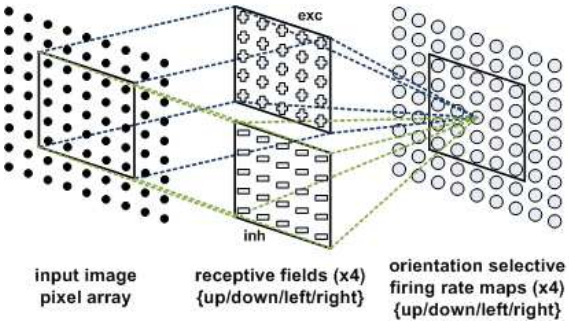

Figure 3: SNN architecture for edge detection (Glackin et al., 2009a).

nected to the layer (x,y) while being combined with the neighbouring pixels via the 5x5 matrix.

In order to generate the final output layer which represents the output image, each intermediate layer $N_{1...4}$ is overlaid. Thus, each pixel in the final output layer represents the number of times that each of the neurons in an intermediate layer produced an output spike during the simulation period.

This section of the paper has given the reader a general overview of the SNN edge detection architecture which has been implemented for comparison purposes. A significantly more detailed and comprehensive description of the SNN application has been presented (Wu et al., 2007).

4 RESULTS

The conductance based integrate and fire model for spiking neurons has been used. For a detailed description of the conductance based model the reader is referred to (Maguire et al., 2007).

Figure 4: Input image.

In order to test the GPU implementation against the FPGA and CPU software implementations it was necessary to select the same input image as has been used previously ie the well known, 'Lena' image. The actual 'Lena' image shown in figure 4 which was used has a resolution of $512x512$ pixels, which when applied to the SNN edge detection application requires $1.05x10^6$ conductance based integrate and fire neurons and $52.4x10^6$ synapse's.

```
Begin
    Load input image
    Determine image size
    Convert pixel values
    Apply RF field
    Allocate and initialise memory on GPU
    Copy to the GPU
    For each timestep do,
        <<< Execute Orientation
                        Selective kernels >>>
    Execute Output stage kernel
    Copy output data from the GPU
    Free GPU memory
    Save output image
End
```

Outlined above is the basic implementation flow for the SNN application. As can been seen there are a number of steps which will only be executed once during the lifetime of the program and in terms of

time spent implementing versus performance gained, they where not considered as initial targets for the optimization effort. However, as the intermediate (Orientation Selective) section of the code is executed at every time step and the spiking neurons can typically take advantage of parallel platforms, this element of the algorithm was deemed a good target for the optimization effort. For each of the eight orientation selective maps discussed in section 3, a CUDA kernel was implemented. The SNN application was then profiled on a standard desktop PC which contained an Nvidia C1060 card with 4GB of DDR3 memory and 240 scalar processors. As shown in Table 1 the GPU was able to provide a 23.49 times speed increase over CPU version and a 1.37 times increase over the FPGA version.

Table 1: SNN simulation times using 512*x*512 image.

Implementation	Computation time	Speed up
Nvidia C1060	5.109 Secs	n / a
FPGA Platform	7 Secs	1.37x
Intel Xeon 2.6 Ghz	120 Secs	23.49x

Figure 5: Output image.

The approach used to validate the GPU based implementation was to compare the output image shown in figure 5 with the equivalent image produced by the FPGA and CPU implementations. When the output images where compared both visually and programmatcially they where found to be identical.

Additionally once the implementations where evaluated and found to produce identical output results, it was decided to increase the number of neurons which would be simulated. However, due to memory limitations on the FPGA it wasn't possible to simulate the increased network sizes. Figure 6 shows the simulation runtime times as the number of neurons was increased from approximately 1 Million to 16.8 Million. As can deduced from the graph even with the GPU simulating 16.8 million neurons, the simulation time is still only around 65% of the time required for 1 million neurons on a CPU.

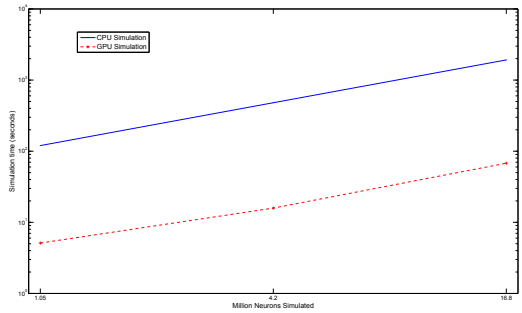

Figure 6: CPU v GPU simulation times.

5 CONCLUSIONS

The edge detection algorithm described in (Wu et al., 2007) has been reimplemented using a C1060 based card from the NVIDIA family of graphic processing units, using the CUDA API as the programming model. The NVIDIA GPU approach has been show to provided a speed-up of 23.49 times over the Intel Xeon CPU based implementation as reported (Glackin et al., 2009a) and a performance improvement of 1.37 times when compared with the novel custom FPGA platform reported in (Glackin et al., 2009a).

Further work will investigate further reducing the runtime while increasing the size of the image that can be processed and thus increasing the number of neurons required for implementation, as well as exploring both improved SNN architectures and more complex neuron models such as the Hodgkin Huxley model (Gerstner and Kistler, 2002) and the Izhikevich model (Izhikevich, 2003). The more complex biologically plausible models have not been as widely used within the bio-inspired application area, due to the computationally expensive nature of their implementation. It is hoped that the massive computational power provided by the Tesla Nvidia GPU processing platform can be leveraged to allow engineers a greater range of exploration within this challenging and exciting research area.

ACKNOWLEDGEMENTS

This research is supported under the Centre of Excellence in Intelligent Systems (CoEIS) project, funded by the Northern Ireland Integrated Development Fund and InvestNI.

REFERENCES

Gerstner, W. and Kistler, W. M. (2002). *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press.

Glackin, B., Harkin, J., McGinnity, T., Maguire, L., and Wu, Q. (2009a). Emulating Spiking Neural Networks for Edge Detection on FPGA Hardware. *isrc.ulster.ac.uk*.

Glackin, B., Harkin, J., McGinnity, T. M., and Maguire, L. P. (2009b). A Hardware Accelerated Simulation Environment for Spiking Neural Networks. In *Proceedings of 5th International Workshop on Applied Reconfigurable Computing (ARC'09)*, volume 5453 of *Lecture Notes in Computer Science*, pages 336–341.

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, 14:1569–1572.

Khan, M., Lester, D., Plana, L., Rast, A., Jin, X., Painkras, E., and Furber, S. (2008). SpiNNaker: mapping neural networks onto a massively-parallel chip multiprocessor. In *Proc. 2008 Intl Joint Conf. on Neural Networks (IJCNN2008)*, pages 2849–2856.

Kirk, D. B. and Hwu, W.-m. W. (2010). *Programming Massively Parallel Processors*. Elsevier.

Maguire, L. P., McGinnity, T. M., Glackin, B., Ghani, A., Belatreche, A., and Harkin, J. (2007). Challenges for large-scale implementations of spiking neural networks on FPGAs. *Neurocomputing*, 71:13–29.

Nageswaran, J. M., Dutt, N., Krichmar, J. L., Nicolau, A., and Veidenbaum, A. (2009). Efficient simulation of large-scale spiking neural networks using CUDA graphics processors. In *International conference on neural networks*.

Wu, Q., McGinnity, M., Maguire, L., Belatreche, A., and Glackin, B. (2007). Edge Detection Based on Spiking Neural Network Model. In *International Conference on Intelligent Computing*, pages 26–34. Springer Verlag.