BLOCK-BASED NEURAL NETWORK MAPPING ON GRAPHICS PROCESSOR UNIT

ONG CHIN TONG

UNIVERSITI TEKNOLOGI MALAYSIA

BLOCK-BASED NEURAL NETWORK MAPPING ON GRAPHICS PROCESSOR UNIT

ONG CHIN TONG

A project report submitted in partial fulfilment of the
requirements for the award of the degree of
Master of Engineering (Electrical-Computer and Microelectronic System)

Faculty of Electrical Engineering
Universiti Teknologi Malaysia

JUNE 2015

Dedicated, in thankful appreciation for support, encouragement and understanding to my beloved mother, father, brother and supervisor....

ACKNOWLEDGEMENT

First and foremost, thank God for giving me the strength to complete this project. I wish to express my greatest appreciation to my supervisor, Assoc Prof Dr. Muhammad Nadzir bin Marsono for his generous guidance, advice and motivation throughout this study. His critical suggestion and comments help me to develop the study project into understandable and workable effort.

Besides that, I would also like to thank Dr Vishnu P. Nambiar from Intel for providing me the suggestion of this topic. Thank you for willing to allocate his time to guide me throughout my works.

Finally, acknowledgements are also for my family, especially my parents. Thank you for their support, inspiration, and encouragement during the study and completion of this thesis.

Ong Chin Tong

ABSTRACT

Block-based neural network (BbNN) was introduced to improve the training speed of artificial neural network. Various works had been carried out by previous researchers to improve training speed of BbNN system. Multithread BbNN training on field-programmable gate array (FPGA) limits training speed due to low performance of Nios II software used for communication between central processing unit (CPU) and FPGA. This project aims to improve training speed of multithread BbNN block by mapping BbNN model into Compute Unified Device Architecture (CUDA) core. In this project, each BbNN block is mapped into a CUDA core with each core running on a single thread. The functional verification of BbNN core is carried out based on the BbNN output accuracy value. Near 100 percent accuracy value obtained is used to verify the CUDA mapped BbNN. The performance trade-off analysis had been carried out by comparing the accuracy value obtained from BbNN evolution on GPU versus CPU implementations. From the results obtained, it is found out that the performance of CUDA-mapped BbNN can only be as fast as CPU-mapped implementation. Although CUDA-mapped BbNN implementation run multiple BbNN blocks training in parallel, large data transfer between CPU and GPU dominates the performance gain in training multiple BbNN blocks in parallel. Besides that, a significant gain in training speed can only be seen if the order of complexity for GPU execution is at a higher order compared to the order of CPU-GPU data transfer. The result obtained in this project provides recommendation for future research works on how to further improve the training speed of CUDA-base BbNN implementation.

ABSTRAK

Block rangkaian neural (BbNN) telah diperkenalkan untuk menyingkatkan masa pemprosesan rangkaian neural. Pelbagai kerja telah dijalankan oleh penyelidik sebelum ini untuk menyingkatkan masa pemprosesan BbNN. Prestasi multithread BbNN menggunakan field-programmable gate array (FPGA) akan dihadkan oleh prestasi perlahan daripada perisian Nios II yang digunakan untuk berkomunikasi antara central processing unit (CPU) dan FPGA. Projek ini bertujuan untuk menerokai kaedah bagi menyingkatkan masa pemprosesan dengan memetakan BbNN mengunakan teras Compute Unified Device Architecture (CUDA). Dalam projek ini, setiap blok BbNN dipetakan ke dalam teras CUDA dengan setiap teras berjalan dengan satu thread. Dengan mendapat ketepatan yang hampir kepada 100 peratus, BbNN yang dipetakan ke dalam CUDA telah disahkan betul. Perbandingan antara prestasi GPU dan CPU kemudian dijalankan dengan mendapatkan perbezaan ketepatan dan masa pemprosesan BbNN. Daripada keputusan projek ini, didapati kelajuan pemprosesan BbNN yang dipetakan ke dalam teras CUDA hanya seiras dengan masa pemprosesan BbNN CPU. Walaupun BbNN yang dipetakan ke dalam teras CUDA diprocess secara selari, prestasi masa pemprosesan CUDA telah didominasi oleh jumlah besar data yang perlu disampaikan antara CPU dan GPU. Di samping itu, peningkatan prestasi pemprosesan CUDA hanya dapat diperlihat sekiranya kerumitan pembilangan berada dalam order yang lebih tinggi daripada kerumitan data yang perlu disampaikan. Keputusan yang diperolehi daripada projek ini dapat menyediakan cadangan untuk kajian masa hadapan mengenai cara untuk meningkatkan lagi prestasi BbNN yang dipetakan ke dalam teras CUDA.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Problem Background

Block-based neural network (BbNN) was introduced by Merchant and Kong in 2001 [1]. It is composed of regular networks of neuron blocks that are connected in a 2-dimensional grid manner. BbNNs are generally used for classification problems [2]. Each block in the network structure is a basic processing element which consists of four input/output nodes. BbNN has modular structure that allows it to be scaled easily according to the complexity of the problem in hand. This can be done by modifying the number of rows and columns of BbNN structure. BbNN internal configuration can be varied during training according to the problem encounter as the training is carried out using evolutionary algorithm such as genetic algorithm.

There are various techniques used to model BbNN model. This includes general purpose processor (CPU)-based BbNN as well as field-programmable gate array (FPGA)-based BbNN [2]. Implementation of BbNN on FPGA is suitable as FPGA has similar structure as compared to BbNN. Each FPGA internal Logic Array Block (LAB) can be directly mapped to BbNN. Although BbNN implementation using FPGA seems to be providing a relatively promising performance, there still some room for improvement in terms of its implementation performance as well as its usability.

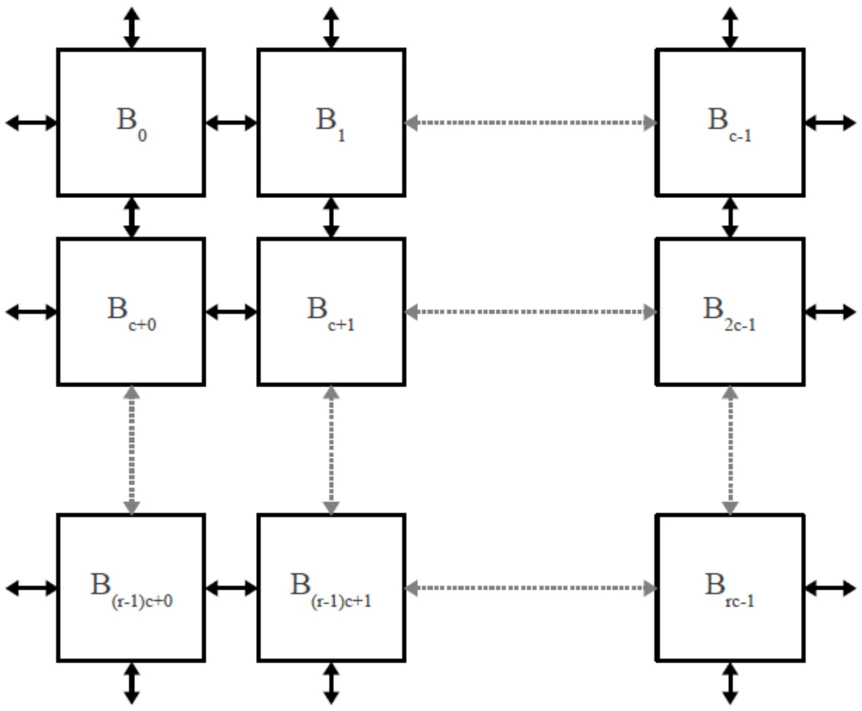

Figure 1.1: General Structure of BbNN [2].

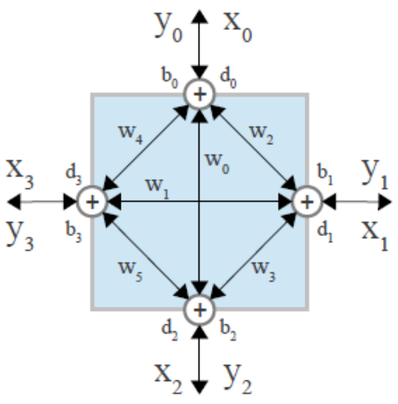

Figure 1.2: Typical Structure of a Single Neuron [2].

1.2 Problem Statement

From previous work done by Nambiar et al. [2], it was known that BbNN implementation using FPGA was limit by the performance of Nios II software. They suggested that a better solution could have been made by implementing BbNN using a faster embedded processor. This project explores an alternative solution to the slow training speed of FPGA BbNN implementation by implementing BbNN structure using embedded Graphic Processor Unit (GPU). It is because same processing can be directly mapped to a neuron block of BbNN. Besides that, the matured Nvidia Cuda programming language that was initially developed for gaming purposes will not become the bottleneck for the BbNN implementation. This can be proven from the positive feedback of smooth gaming experience from gamers all around the world.

1.3 Objectives

The objective of this project is to propose a mapping technique that maps BbNN into GPU based system using Nvidia CUDA programming language. The functionality of the proposed implementation will be verified through simple XOR logic calculation. Meanwhile the design trade-off of the proposed BbNN mapping technique will be analyzed as compared to previous CPU and FPGA implementations based on Tomita classification problem.

1.4 Scope

The BbNN code to be used for GPU mapping in this project would be the one developed by Nambiar et al. [2]. The Genetic algorithm used will be remained unchanged. This means that DemeGA will be used as the Genetic algorithm for BbNN training throughout this project. Besides, the size of BbNN will also follows what Nambiar et al. has in their previous implementation, which is one row and ten columns. The number of maximum BbNN populations generation would be 5000 as this was the number chosen by Nambiar et al. in their previous implementation. Meanwhile, the cross-over and mutation rate will be using a default value of 0.35 and 0.006 respectively.

1.5 Methodology

Nvidia GeForce 840M GPU, which utilizing Nvidia's latest Maxwell architecture is used for the proposed BbNN mapping technique. All development work is done under Ubuntu Linux environment using Nvidia Cuda programming language.

The methodology of this project starts with a detailed analysis and study about the architecture of BbNN implementation on Nvidia GPU. This is to sought out the way for BbNN mapping implementation. In order to do so, readings on previous BbNN implementation structure was carried out. This structure was then compared to the internal architecture of Nvidia GPU.

After a thorough study about the architecture, a mapping technique formed and implemented on Nvidia GPU using Nvidia Cuda programing language. The parameter used for BbNN implementation using GPU should be the same as previous FPGA implementation [2] in order for fair performance comparison analysis.

This project then proceeds with functionality verification using Tomita training database [2]. At the end of this project, the performance analysis of the proposed GPU BbNN implementation is carried out using same Tomita classification problem and compared against CPU and FPGA BbNN implementations.

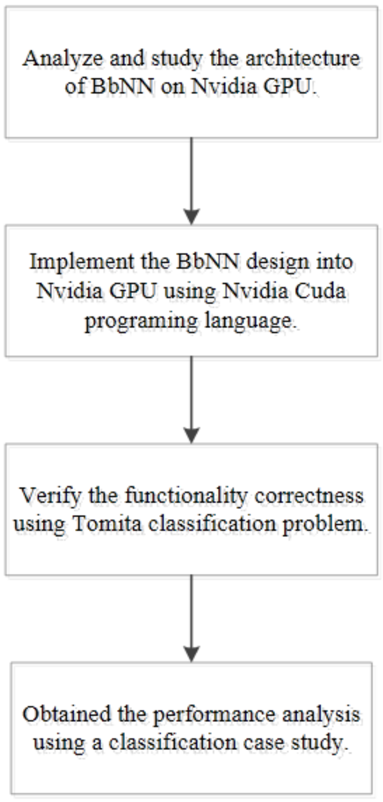
Analyze and study the architecture of BbNN on Nvidia GPU.
Implement the BbNN design into Nvidia GPU using Nvidia Cuda programing language.
Verify the functionality correctness using Tomita classification problem.
Obtained the performance analysis using a classification case study.

Figure 1.3: Project Methodology.

1.6 Report Organization

Figure 1.4 shows the report organization. This report is organized into six chapters. The rest of the report is organized as follows. In Chapter 2, this report first go through all available BbNN architectures as well as its mathematical structure, Genetic Algorithm that used for BbNN training, how BbNN used in solving classification problems, related works and the motivation for this project. Chapter 3 covers the proposed BbNN implementation where the details of the design process and requirements will be discussed. In Chapter 4, this report discuss on the implementation of BbNN. Besides, details on the verification of the proposed BbNN implementation were discussed in this chapter. Chapter 5 includes analyses on the design trade-off of the proposed BbNN implementation. Meanwhile chapter 6 concludes the report and point out the direction of future work.

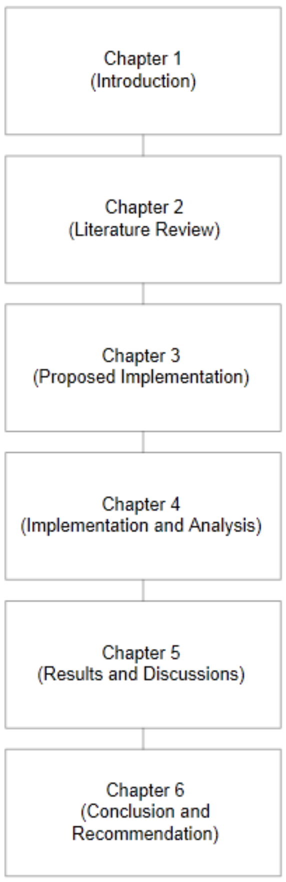
Chapter 1
(Introduction)
Chapter 2
(Literature Review)
Chapter 3
(Proposed Implementation)
Chapter 4
(Implementation and Analysis)
Chapter 5
(Results and Discussions)
Chapter 6
(Conclusion and Recommendation)

Figure 1.4: Report Organization.

REFERENCES

1. S. W. Moon, S. G. K. Block-based neural networks. *IEEE Transactions on Neural Networks*, 2001: 307–17.
2. Nambiar, V. P. Hardware Implementation of Evolvable Block-based Neural Networks Utilizing a Cost Efficient Sigmoid-like Activation function. *Neurocomputing*, 2014: 228–241.
3. Nambiar, B. S. K.-H. M., V. P. and Marsono, M. N. HW/SW co-design of reconfigurable hardware-based genetic algorithm in FPGAs applicable to a variety of problems. *Computing*: 95(9), 863–896.
4. Samarah, H. A. T. S.-K. N., A. Automated coverage directed test generation using a cell-based genetic algorithm. *Eleventh Annual IEEE International*, 2006: 19–26.
5. Kothandaraman, S. *Implementation of Block-based Neural Networks on Reconfigurable Computing Platforms*. M.sc. thesis. University of Tennessee. 2004.
6. Merchant, S. G. and Peterson, G. D. Evolvable Block-based Neural Network Design for Applications in Dynamic Environments. 2010.
7. Jewajinda, Y. An Adaptive Hardware Classifier in FPGA Based-on a Cellular Compact Genetic Algorithm and Block-based Neural Network. 2008. 658–663.
8. Brown, M. Nvidia Unveils Maxwell: A Supremely Power-efficient GPU Architecture. URL `http://www.pcworld.com/article/2097974/a-supremely-power-efficient-gpu-architecture.html`.
9. Tesla. What Is Gpu Accelerated Computing. URL `http://www.nvidia.com/object/what-is-gpu-computing.html`.
10. Wikipedia. CUDA. URL `http://en.wikipedia.org/wiki/CUDA`.
11. Stackoverflow. In CUDA, why cudaMemcpy2D and cudaMallocPitch consume a lot of time. URL `http://stackoverflow.com/questions/24280220/`

why-cudamemcpy2d-cudamallocpitch-consume-a-lot-time.

12. Sanders, . K. E., J. CUDA by example: an introduction to general-purpose GPU programming. *Addison-Wesley Professional*.

13. Nvidia. NVIDIA. CUDA Toolkit 5.0 Performance Report. URL http://on-demand.gputechconf.com/gtc-express/2013/presentations/cuda--5.0-math-libraries-performance.pdf.

14. Buck, F. K. . H.-P., I. GPUBench: Evaluating GPU performance for numerical and scientific applications. *In Proceedings of the 2004 ACM Workshop on General-Purpose Computing on Graphics Processors*.

15. amazon. Intel Core i7-990X Extreme Edition Processor 3.46 GHz 6 Core LGA 1366 - BX80613I7990X. URL http://www.amazon.com/Intel-i7-990X-Extreme-Edition-Processor/dp/B004NRQDQQ.

16. tom'sHARDWARE. Core i7 GFLOPS Benchmark. URL http://www.tomshardware.com/forum/262886-28-core-gflops-benchmark.

17. Jewajinda, Y. An adaptive hardware classifier in FPGA based-on a cellular compact genetic algorithm and block-based neural network. *International Symposium on Communications and Information Technologies (ISCIT 2008)*, 2008: 658–663.

18. Jewajinda, Y. and Chongstitvatana, P. FPGA-based online-learning using parallel genetic algorithm and neural network for ECG signal classification. *International Conference on Electrical Engineering/Electronics Computer Telecommunications and Information Technology (ECTICON)*, 2010: 1050–1054.

19. Jewajinda, Y. and Chongstitvatana, P. A parallel genetic algorithm for adaptive hardware and its application to ECG signal classification. *Neural Computing and Applications*, 2012: 1069–1626.

20. V. P. Nambiar, M. M., M. Khalil-Hani. Evolvable Block-Based Neural Networks for Real-Time Classification of Heart Arrhythmia From ECG Signals. *IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, 2012: 866–871.

21. V. P. Nambiar, C. S. M. M., M. Khalil-Hani. Evolvable blockbased neural networks for classification of driver drowsiness based on heart rate variability.

IEEE International Conference on Circuits and Systems (ICCAS), 2012: 156–161.

22. M. Khalil-Hani, N. S.-H. V. P. N., C.W. Sia. FPGA-based Embedded System for the detection of Driver Drowsiness using ECG signals. *Proceedings of the 2012 International Conference on Electrical Engineering and Computer Science (ICEECS 2012)*, 2012.

23. W. Jiang, G. P., S. Kong. ECG signal classification using blockbased neural networks. *IEEE International Joint Conference on Neural Networks (IJCNN05)*, 2005: 326–331.

24. Goldberg, D. E. Genetic Algorithms in Search, Optimization, and Machine Learning. *Addison-Wesley Professional*, 1989.

25. M. Nickray, A. A.-k., M. Dehyadgari. Power and delay optimization for network on chip. *Proceedings of the 2005 European Conference on Circuit Theory and Design, IEEE*, 2005.

26. K. De Jong, W. S. Using genetic algorithms to solve np-complete problems. *Proceedings of the third international conference on genetic algorithms*, 1989: 132.

27. W. Jiang, S. K. A Least-Squares Learning for Block-based Neural Networks, Advances in Neural Networks. *A Supplement (DCDIS)*, 2007: 242–247.

28. S. Haridass, D. H. Fault Tolerant Block Based Neural Networks. *System Theory (SSST)*, 2010: 357–361.

29. Q. A. Tran, J. H., F. Jiang. A Real-Time NetFlow-based Intrusion Detection System with Improved BBNN and High-Frequency Field Programmable Gate Arrays. *IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (Trust-Com)*, 2012: 201–208.

30. A. R. Brodtkorb, M. L. S., T. R. Hagen. Graphics processing unit (GPU) programming strategies and trends in GPU computing. *Journal of Parallel and Distributed Computing*, 2013: 4–13.