FIELD PROGRAMMABLE GATE ARRAY BASED CONVOLUTION NEURAL NETWORK HARDWARE ACCELERATOR WITH OPTIMIZED MEMORY CONTROLLER

MOHAMMED ISAM ELDIN HASSAN MOHAMMED

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

School of Electrical Engineering
Faculty of Engineering
Universiti Teknologi Malaysia

JULY 2020

DEDICATION

This project report is dedicated to my parents, who have given me a lifetime of love, and care. It is also dedicated to my sister and two brothers for their unlimited support and for keeping my spirit up.

ACKNOWLEDGEMENT

I wish to express my deepest gratitude to my supervisor, Doctor Mohd Shahrizal Bin Rusli, who has the substance of a genius: he convincingly guided and encouraged me to be professional and do the right thing even when the road got tough. Without his persistent help, the goal of this project would not have been realized.

Also, I would like to acknowledge the support and great love of my family, my father, Isam Eldin; my mother, Magda Abdelrahem; my sister, Alya; and my two brothers, Wadah and Hassan. They kept me going on and this work would not have been possible without their input.

ABSTRACT

Convolution Neural Network (CNN) is a special kind of neural network that is inspired by the behaviour of optic nerves in living creatures. CNN is gaining more and more attention nowadays because of the increased demand for high speed and low-cost synthetic vision systems. However, CNN can be both compute- and memory-intensive. For that reason, implementation in a general-purpose processor will be slow and inefficient. Therefore, this project proposes a flexible CNN hardware accelerator that targets the Field Programmable Gate Array (FPGA) platform and features an optimized memory controller to reduce redundancy memory access. The main advantage of this project is that the accelerator is flexible - meaning that the user of the accelerator has the capability of modifying the architecture using parameterization to optimize for execution speed, resource utilization, and power consumption. The accelerator employs various hardware design techniques like loop unrolling, pipelining, optimized memory controller, and others to achieve the targeted performance. The accelerator is written in System Verilog language using Xilinx's Vivado software and is tested using a single convolution layer from several selected CNN architectures. Then, it is compared against the same convolution layer implemented in Matlab. The proposed accelerator shows a huge speedup compared to the software counterpart of up to 4251X speed up with reasonable resource utilization and consumes only 0.27 W per layer.

ABSTRAK

Convolution Neural Network (CNN) adalah sejenis rangkaian saraf khas yang diilhamkan oleh tingkah laku saraf optik pada makhluk hidup. CNN semakin mendapat perhatian sekarang kerana permintaan yang tinggi untuk sistem penglihatan sintetik berkelajuan tinggi dan kos rendah. Walau bagaimanapun, CNN boleh memerlukan banyak komputasi dan memori. Untuk itu, pelaksanaan pada pemproses serba guna akan menjadi lambat dan tidak cekap. Oleh itu, projek ini mencadangkan pemecut perkakasan CNN fleksibel yang mensasarkan platform Field Programmable Gate Array (FPGA) dan dilengkapi pengawal memori yang dioptimumkan untuk mengurangkan akses memori berlebihan. Kelebihan utama projek ini adalah bahawa pemecut yang fleksibel - bermaksud bahawa pengguna pemecut memiliki kemampuan mengubah seni bina menggunakan pemparameteran untuk mengoptimumkan kecepatan pelaksanaan, penggunaan sumber daya dan penggunaan daya. Pemecut menggunakan pelbagai teknik reka bentuk perkakasan seperti membuka gelung, penalian paip, pengendali memori yang dioptimumkan dan lain-lain untuk mencapai prestasi yang disasarkan. Pemecut ditulis dalam bahasa Sistem Verilog menggunakan perisian Vivado Xilinx dan diuji menggunakan lapisan perlingkaran tunggal dari sebilangan seni bina CNN terpilih. Kemudian, ia dibandingkan dengan perlingkaran yang sama lapisan dalam Matlab. Pemecut yang dicadangkan menunjukkan kelajuan yang besar berbanding dengan perisian yang sehingga 4251X dengan penggunaan sumber yang wajar dan hanya menggunakan 0.27 W per lapisan.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CNN	-	Convolution Neural Network
RELU	-	Rectified Linear Units
HDL	-	Hardware Description Language
FPGA	-	Field Programmable Gate Array
GPU	-	Graphical Processing Unit
MLP	-	Multi-Layer Perceptron
PE	-	Processing Elements
BRAM	-	Block Random Access Memory
MAC	-	Multiply Accumulate Unit
LUT	-	Look-Up Table
RTL	-	Register Transfer Level
UAV	-	Unmanned Aerial Vehicle
ILSVRC	-	ImageNet Large Scale Visual Recognition Competition
DSP	-	Digital Signal Processing

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

Convolution neural network (CNN) is a subset of neural networks which is a deep learning artificial intelligence technique that is inspired by the architecture of the human brain, a neural network consists of multi-layer neuron connections that function together to achieve high accuracy in classification and recognition tasks. The difference between ordinary neural networks and CNNs is that the latter feature multiple layers of convolution operation that is used to take advantage of data locality in images to reduce the number of parameters needed to execute large images in a conventional neural network [1].

CNN is very popular and widely used in synthetic vision systems compared to other artificial intelligence techniques/algorithms. The reasons for that are firstly in many machine learning methods the feature extraction step and classification step are separate, both needed to be implemented differently. Meanwhile, in CNN feature extraction and classification is done in one step. Secondly, the success of the vision system heavily depends on a successful feature extraction process which is usually done manually and needs a lot of tweaks to achieve the required performance in a non-CNN method which is a very time-consuming task [2, 3]. The last reason is that CNN is fast scaling-up networks which make it easier and faster to take an existing network architecture and scale it up to the desired system [4].

CNN is gaining more attention nowadays because of the increased demand of high-speed low-cost synthetic vision systems that are used to identify and categorize different objects in an image, examples of such application are smartphones, imaging sensor network, unmanned air vehicles (UAVs), and other embedded vision applications [1].

The deployment of big CNN models can be both compute-intensive and memory-intensive [4]. For these reasons implementation on general processors will be slow and inefficient. To counter these problems designers usually implement CNN in graphical processing units (GPU), application specific integrated circuit (ASIC), and field programmable gate array (FPGA). GPU implementation has the disadvantages of high power consumption, bulk size, and high price while ASIC implementation is very expensive and hard to customize this lift FPGA as the only reasonable and viable implementation of CNN for embedded systems.

FPGAs have been extensively studied as an important hardware platform for CNN computations. Different from general-purpose architectures, FPGA allows users to customize the functions and organization of the designed hardware in order to adapt various resource needs and data usage patterns [4]. Although current FPGA accelerators have demonstrated better performance over generic processors, the accelerator design space has not been well exploited. One critical problem is that the computation throughput may not well match the memory bandwidth provided by the FPGA platform [5].

According to previous studies [6, 7] convolutional layer account for over 90% of the computation of the total CNN execution time. Another issue highlighted by the study [8] is the challenging problem of memory bottleneck and the need to use a flexible memory hierarchy that supports the complex data access patterns on CNN. This work addresses these two problems by introducing a hardware accelerator that supports the convolution operation and features a memory controller that reduces the redundancy memory accesses, the accelerator will be designed using system Verilog hardware descriptive language (HDL) targeting FPGA implementation.

1.2 Problem Statement

Many solutions have been proposed to overcome the software implementation limitations of CNN in general processors. Among the solutions are GPUs and ASICs. Indeed, systems that utilize these platforms can achieve better performance in terms of execution time. However, the GPU implementation is energy-intensive (having high power consumption) and the ASIC initial cost is very high and also it is inflexible.

As said earlier according to previous studies [6,7] convolutional layer account for over 90% computation of the total CNN execution time. This increases the demand for dedicated hardware acceleration for this convolution operation.

While the software implementation of CNN in GPUs gives high accuracy, it cannot meet real-time embedded systems constraints, such as power consumption, and cost. In addition to that, in the software implementation, there are unnecessary redundant memory accesses which slow down the overall execution time.

FPGA allows the designer to customize the functions and organization of the designed hardware logic in order to adapt various resource needs and data usage patterns. It also allows the designer to optimize for either execution time, power consumption, or design area.

1.3 Research Objectives

The objectives of this project are:

(a) To design a parameterized hardware accelerator for CNN using system Verilog language that is capable of:

   1. Accelerating the convolution operation.
   2. Supporting Relu as a nonlinear activation function.
   3. Supporting Max pooling for the pooling layer.

(b) To design a memory controller that decreases the redundancy in data accessing to improve execution time.

(c) To compare the performance of the convolution operation in the hardware accelerator against its software equivalence in terms of execution time.

(d) To synthesize different architectures of the accelerator and compare them in terms of execution time, resource utilization, and power consumption.

1.4 Research Scope

To explore the hardware optimization techniques and design a hardware accelerator that supports and accelerates the common function required in modern CNN such as convolution operation, nonlinear activation function, and pooling operation using system Verilog HDL. The accelerator support only Relu activation function as it is the most popular nonlinearity function in today's CNN, other activation function such as sigmoid, tanh are not supported. For the pooling layer, only max pooling is supported.

For testing and validating the design, a single convolution layer from several selected CNN architectures is implemented and is compared against the same convolution layer implemented in Matlab. For the testing data, random data is used for the input image, bias data, and kernel data. the same random data is fed to the hardware accelerator and the software and the results are compared to validate the design.

1.5 Contribution

Deep neural networks are used in synthetic vision systems because of their versatility and as such, are suitable for a variety of vision tasks. A low power consumption FPGA based accelerator give a low powered embedded system such as UAVs, security monitoring, smartphones and other synthesis vision application the capability of achieving high performance and accuracy in term of image classification and recognition.

1.6 Chapters Organization

Chapter 2 presents the literature reviews. It contains the introduction of CNN, discussion about some of the popular CNN architectures, and various state of the art CNN hardware accelerators are discussed, compared and reviewed.

Chapter 3 describes the methodology. The overall project flow is presented. And the general architecture of the accelerator is discussed. This chapter also presents various hardware optimization technique that is used to achieve the final design such as loop unrolling, pipeline, parameterization, and others.

Chapter 4 presents a detailed view of the accelerator building blocks which include register transfer level (RTL) flow charts and circuit diagrams. The results of the optimized memory controller and the complete hardware accelerator are also included in this chapter. Finally, Chapter 5 concludes the project and future works are discussed.

REFERENCES

1. C. Farabet, B. Martini, P. Akselrod, S. Talay, Y. LeCun, and E. Culurciello, "Hardware accelerated convolutional neural networks for synthetic vision systems," in *ISCAS*, 2010, vol. 2010, pp. 257-260.

2. Z. Liu, Y. Dou, J. Jiang, Q. Wang, and P. Chow, "An FPGA-based processor for training convolutional neural networks," in 2017 International Conference on Field Programmable Technology (ICFPT), 2017, pp. 207-210: IEEE.

3. N. Aloysius and M. Geetha, "A review on deep convolutional neural networks," in 2017 International Conference on Communication and Signal Processing (ICCSP), 2017, pp. 0588-0592: IEEE.

4. S. Li, W. Wen, Y. Wang, S. Han, Y. Chen, and H. Li, "An FPGA design framework for CNN sparsification and acceleration," in 2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2017, pp. 28-28: IEEE.

5. C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2015, pp. 161-170: ACM.

6. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, 2012, pp. 1097-1105.

7. J. Cong and B. Xiao, "Minimizing computation in convolutional neural networks," in International conference on artificial neural networks, 2014, pp. 281-290: Springer.

8. M. Peemen, A. A. Setio, B. Mesman, and H. Corporaal, "Memory-centric accelerator design for convolutional neural networks," in 2013 IEEE 31st International Conference on Computer Design (ICCD), 2013, pp. 13-19: IEEE.

9. S. Chakradhar, M. Sankaradas, V. Jakkula, and S. Cadambi, "A dynamically configurable coprocessor for convolutional neural networks," in Proceedings of the 37th annual international symposium on Computer architecture, 2010, pp. 247-257.

10. T. Chen et al., "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," ACM SIGARCH Computer Architecture News, vol. 42, no. 1, pp. 269-284, 2014.

11. V. Gokhale, J. Jin, A. Dundar, B. Martini, and E. Culurciello, "A 240 g-ops/s mobile coprocessor for deep neural networks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2014, pp. 682-687.

12. T. Y. chin, "TONGUE COLOR DIAGNOSIS USING DEEP LEARNING TECHNIQUE," Master thesis, 2019.

13. K. Hornik, "Approximation capabilities of multilayer feedforward networks," Neural networks, vol. 4, no. 2, pp. 251-257, 1991.

14. S. S. Liew, M. Khalil-Hani, and R. Bakhteri, "Bounded activation functions for enhanced training stability of deep neural networks on visual pattern recognition problems," Neurocomputing, vol. 216, pp. 718-734, 2016.

15. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.

16. C. Szegedy et al., "Going deeper with convolutions," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1-9.

17. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770-778.

18. S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in Advances in neural information processing systems, 2015, pp. 1135-1143.

19. J. L. Holi and J.-N. Hwang, "Finite precision error analysis of neural network hardware implementations," IEEE Transactions on Computers, vol. 42, no. 3, pp. 281-290, 1993.