DESIGN OF A NEURAL NETWORK FOR FPGA IMPLEMENTATION

LIM EE RIC

UNIVERSITI TEKNOLOGI MALAYSIA

DESIGN OF A NEURAL NETWORK FOR FPGA IMPLEMENTATION

LIM EE RIC

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 2013

ABSTRACT

Very often complex transfer functions are needed to be implemented in ASIC for faster or real-time application. Other than implementing a transfer function according to its equation or algorithm, prediction method can be used in certain application where accuracy can be tolerated. In this project, application of neural network as a predictor is studied. Focus will be placed on back-propagation feed-forward neural network and its realization in hardware using Verilog Hardware Descriptive Language (HDL). Hardware design challenges like hardware resource utilization, throughput of various design approaches were explored. Main objective of this project is to produce a high throughput reconfigurable back propagation neural network hardware module that can be applied or integrated into bigger hardware system. Altera Quartus II and ModelSim-Altera CAD tool was used as logic synthesizing tool and hardware simulation tool, respectively, to achieve abovementioned objective. MATLAB was also being used to model neural network in software which served as a benchmark for hardware design. Multi-cycle design approach successfully reduces resource utilization on hardware-intensive neural network module, while pipelining the design helped to achieve a high-throughput design. Utilization of RAM for reconfiguration purpose greatly reduced throughput of the design due to the fact that only one weight or bias values are loaded in every clock cycle.

ABSTRAK

Seringkali persamaan matematik yang rumit perlu direalisasikan di ASIC dengan tujuan untuk meningkatkan prestasi pengiraan. Sekiranya applikasi boleh bertolak ansur dengan ketepatan yang tidak begitu tinggi, maka selain daripada melaksanakan pengiraan matematik dengan mengukuti algoritmanya, kaedah ramalan boleh digunakan. Dalam projek ini, rangkaian neural ataupun *neural network* telah digunakan sebagai medium ramalan untk membuat ramalan bagi sesetengah persamaan matematik yang rumit. Tumpuan telah diberikan kepada salah satu jenis *neural network* yang biasa digunakan iaitu *back-propagation neural network* dan tujuan projek ini adalah untuk menrealisasikan *neural network* ini dengan merakabentuk *neural network* dengan Verilog HDL. Cabaran daripada projek reka bentuk init seperti penggunaan sumber, prestasi reka bentuk telah dikaji. Objective utama projek ini adalah untik meraka bentuk *neural network* yang berprestasi tinggi and dapat menghasilkan pengiraan dalam masa yang singkat. Aplikasi Altera Quartus II dan ModelSim-Altera CAD telah digunakan dalam proses reka bentuk. Selain daripada itu, MATLAB juga digunakan untuk mengira and mengimulasi ramalan *neural network* supaya jawapan daripada MATLAB boleh digunakan sebagai rujukan kepada rake bentuk projek ini. Kaedah *Multi-cycle* telah digunakan dalam projek ini untuk mengurangkan penggunaan sumbar reka bentuk. *Pipelining* pula digunakan untuk meningkatkan prestasi reka bentuk supaya neural network dapat manghasilkan jawapan yang lebih banyak dalam masa yang singkat.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ANN	-	Artificial Neural Network
FPGA	-	Field Programmable Gate Array
ASIC	-	Application Specific Integrated Circuit
HDL	-	Hardware Descriptive Language
MATLAB	-	Matrix Laboratory
DSP	-	Digital Signal Processor
PC	-	Personal Computer, or Program Counter
CPU	-	Central Processing Unit
CU	-	Control Unit
LUT	-	Look-up table
GUI	-	Graphical User Interface
CPD	-	Critical Path Delay
MAC	-	Multiply-Accumulate Unit
DU	-	Data path unit
ASM-chart	-	Architectural State Machine Chart
FSM-D	-	Finite State Machine – Data path
BPNN	-	Back-propergation neural network
CS-Table	-	Control Sequence Table

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

Artificial Neural Network (ANN) with its non-linearity characteristic [6] is very powerful in solving many complex computational problems. A series of sequential layers consisting of several simple and similar computational blocks, called neurons are working together in parallel to process output from a given set of inputs, based on weights predicted during training phase. Therefore, even though some complex functions or equations can be solved with the aid of software in general-purpose processor, these problems can be tackled by neural network in a far more optimized and cost-saving manner. This results in high-demand of neural network module as part of a solution to a big problem.

In order take advantage of high-degree of parallelism, hardware implementation of neural network, be it in FPGA or ASIC, often outperform when comparing to general-purpose processor implementation [12]. This is simply because ASIC or FPGA is custom, or semi-custom hardware device, that is optimized based on the application of the network.

1.1 Problem Statement

A large number of hardware architectures have been proposed for the implementation of Artificial Neural Network. Many of them are application specific. For example, utilizing neural network for wind power generation [13], or using ANN to predict a rainfall [3]. Most of the efforts on optimization of hardware ANN architectures have been concentrated on the implementation of the recall phase or "use mode", which is the functional mode of a trained network. The training is often done off-board or off-chip with sophisticated software algorithm on a different platform.

The second type of network is a re-configurable network, where the network is generic or semi-generic for a range of application. This kind of network has far more flexibility, especially to serve as off-the-shelf solution for plug-and-play purposes. However, besides of maintaining the flexibility of re-configuring a network, execution speed and hardware cost is often the major challenge a hardware designer has to balance with, or at least, provide users with the flexibility to determine whether to trade-off execution speed with hardware cost, or vice-versa.

A few architectures will be presented, analyzed, and the pros and cons being compared in chapter *2.2 Literature Review*. Challenges have been identified and key points below are the summarized criteria to be adhered with throughout the entire design project.

1) Reconfigurable network structure: parameterizing number of layers or number of neurons for easier network construction, giving user a choice to opt for sigmoid activation function or linear function, etc.
2) Resource optimized: Reduce resource utilization on a hardware-intensive neural network.
3) Pipeline implementation: Neural network FPGA or ASIC applications usually deal with real-time, high-volume input samples. Pipelining the architecture can definitely take advantage.

1.2 Objective

The objective of this project is to design a reconfigurable high-throughput computational module based on learning algorithm in a neural network on FPGA with Verilog HDL. Besides architecting the building blocks needed to implement the learning algorithm in a neural network, this project includes a series of exploration on various design approach, including one-cycle design method where data flows concurrently from input to output, or multi-cycle designs proper resource planning will be done to reuse or share a specific set of hardware by introducing multi cycle operation, or, in other words, serial data path. Performance of each design will be analyzed and the outcome of the studies and analysis will then lay the foundation for the hardware implementation in Verilog code.

1.3 Scope of Work

Back-propagation feed-forward network is the most popular [6] learning algorithm among all and therefore, the focus of this project will be in creating a back-propagation neural network, which uses a tangent sigmoid as an activation function in its neurons. In addition, with the rule of "always keeping a network simple and small [6]", this project will only focus on a feed-forward neural network, due to the fact that a feed-forward neural network is enough to solve most of the problems.

Besides, generic implementation of a neural network will be as a macro within a bigger design to solve specific problem, thus, learning or training algorithm will not be included within the hardware, instead, can be done off-board with the help of sophisticated simulation software like MATLAB. Only important parameters like the structure of network (e.g. number of layers, number of neurons in each layer, connection between layers, etc.) as well as the weights and biases for each neuron are needed to be loaded into the hardware.

The network module will be catered only for Altera FPGA, using Altera Quartus II as a compiler and simulator. In order to properly implement the network in Altera Quartus II with Verilog HDL, a back-propagation first order neural network has to be generated with sophisticated MATLAB *Neural Network Toolbox*; so that this MATLAB generated simulation result can be taken as a reference for hardware result verification in later design stage. MATLAB generated network will then ported over to synthesizable Verilog code.

Network modules synthesized from Altera Quartus II will then be verified via hardware simulation where ModelSim-Altera CAD tool is used. Test benches will be develop according to features of each design in order to make sure maximum coverage can be achieved. It is worth to note that only hardware simulation is involved in this project, implementing the design on a FPGA board is out of scope of this project.

In addition, focus will also be placed only on prediction application, knowing the fact that neural network can also be applied in some other applications like classification. The network module will be configured to predict a math equation as discussed and tested in latter chapters.

1.4 Research Contribution

A reconfigurable back-propagation feed-forward network module is to be created after completing the entire project. This network module can be integrated or plug-and-play easily to any complex design in order to solve or predict some complex algorithm.

Besides, a proper documentation will be produced after the entire project to document the entire design progress, starting from literature study, up till design verifications, before concluding the project. This documentation will serve as a reference for users who wish to integrate this network module into their design, or future developer who wish to enhance from this project, including implementing this network module in FPGA board.

REFERENCES

[1] Christos Stergiou, Dimitrios Siganos, "*Neural Networks*", Imperial College London.

[2] Ben Krose, Patrick van der Smagt (1996), "*An Introduction to Neural Network*", 8th Ed University of Amsterdam Press, University of Amsterdam,

[3] Kumar Abhishek, Abhay Kumar *et al.* (2012) "*A Rainfall Prediction Model using Artificial Neural Network*", IEEE Control and System Graduate Research.

[4] Robert Lange (2005), "*Design of a Generic Neural Network FPGA-Implementation*.", Professorship of Circuit and Systems Design, Faculty of Elect. Engineering and IT, Chemnitz University of Technology.

[5] Mark Hudson Beale, *et al.* (2012), "*Neural Network Toolbox™ User's Guide"*, Mathworks, MATLAB.

[6] Bogdan M. Wilamowski, "*Neural Network Architectures and Learning Algorithms"*, IEEE Industrial Electronics Magazine, 2009, 56-63

[7] Ng Bee Yee, (2012), "*FPGA Implementation of Image Processing 2D Convolution for Spatial Filter"*, Faculty of Electrical Engineering, Universiti Teknologi Malaysia, Johor.

[8] Ayman Youssef et al. (2012), "*A reconfigurable, Generic and programmable Feed Forward Neural-Network implementation in FPGA*", in 14th International Conf. on Modeling and Simulation, 9.

[9] Gin-Der Wu, et al. (2011), *"Reconfigurable Back Propagation Based Neural Network Architecture"*, in International Symposium on Integrated Circuits.

[10] Mutlu Avci, Tulay Tildirim (2003), "*Generation of Tangent hyperbolic sigmoid function for microcontroller based sigital implementations of neural network*", International XII. Turkish Symposium on Artificial Intelligence and Neural Networks.

[11] Pramod Kumar Meher (2010), "*An Optimized Lookup-Table for the Evaluation of Sigmoid Function for Artificial Neural Networks*", IEEE.

[12] Ramon J. Aliaga, Rafael Gadea, et al. (2009), "*System-on-chip Implementation of Neural network Training on FPGA"*, Int. Journal On Advances in Sysmtems and Measurements, vol 2

[13] Z.S Jiang, D.K Li, et al. (2010), "*PID controller based on BP neural network in the application of wind power generation and matlab simulation",*IEEE.

[14] Y.Wang, J. Du, Z. et al (2011), "*FPGA Based Electronics for PET Detector Modules with Neural network Position Estimators*", IEEE Trans. On Neucler Sc., vol.58, no. 1, 34-42.