AN EVOLVABLE BLOCK-BASED NEURAL NETWORK ARCHITECTURE FOR EMBEDDED HARDWARE

VISHNU A/L PARAMASIVAM

UNIVERSITI TEKNOLOGI MALAYSIA

AN EVOLVABLE BLOCK-BASED NEURAL NETWORK ARCHITECTURE FOR EMBEDDED HARDWARE

VISHNU A/L PARAMASIVAM

A thesis submitted in fulfilment of the
requirements for the award of the degree of
Doctor of Philosophy (Electrical Engineering)

Faculty of Electrical Engineering
Universiti Teknologi Malaysia

JUNE 2013

Dedicated to
my beloved family

ACKNOWLEDGEMENT

First and foremost, I would like to extend my deepest gratitude to my supervisor and teacher, Prof. Dr. Mohamed Khalil Hani, for giving me the opportunity to work in an amazing field of research. His constant encouragement, criticism and guidance were the key to bringing this project to a fruitful completion, especially during the final period of the research. I have learned and gained much, not only in research skills, but also in the lessons of life, which has helped shaped my character. Thanks to him, I now talk and act with better rationale and much gained wisdom. Had we not crossed paths, I would have never realized my full potential.

My sincerest appreciation goes to my co-supervisor Dr. Nadzir Marsono who was always there for me with his cheery attitude, and was of great help academically. Not to forget my seniors Jasmine Hau Yuan Wen and Rabia Bakhteri for their support, help, and technical advices. I have learned much from them, as well as receiving plenty of guidance and motivation.

I would also like to thank all those who have contributed directly and indirectly to the completion of this research and thesis. This includes my fellow postgraduate students who provided me with help and company during my study here. Otherwise, it would have been a lonely journey.

I also want to thank the original developers of the utmthesis LaTeX project for making the thesis writing process a lot easier for me. Thanks to them, I could focus on the content of the thesis, and not waste time with formatting issues.

Finally, I would like to thank my family for always being there for me, through thick and thin. Especially my parents, who are such wonderful role models and respected members of the society. Their role in my life is something I will always need and constantly appreciate.

ABSTRACT

Evolvable neural networks are a more recent architecture, and differs from the conventional artificial neural networks (ANN) in the sense that it allows changes in the structure and design to cope with dynamic operating environments. Block-based neural networks (BbNN) provide a more unified solution to the two fundamental problems of ANNs, which include simultaneous optimization of structure, and viable implementation in reconfigurable embedded hardware such as field programmable gate arrays (FPGAs) due to its modular structure. However, BbNNs still have several outstanding issues to be resolved for an effective implementation. An efficient hardware design can only be obtained with proper design consideration. To date, there has been no previous work reported on BbNNs configured in recurrent mode for complex case studies, even though it is theoretically possible. Existing BbNN models do not explicitly specify or model the latency of the system, determine how it affects the system, nor how it can be optimized. Also, current methods of training BbNNs using genetic algorithm (GA) are slow, especially with large training datasets. This thesis presents an improved BbNN model, proposes a state-of-the-art simulation and co-design environment for it, and implements it on a hardware platform for improved speed and performance. It has a novel architecture with deterministic outputs that can evolve and operate in both feedforward and recurrent modes. The BbNN is redesigned for optimal system latency to achieve higher performance, and supports on-chip training for multi-objective optimization using a multi-population parallel genetic algorithm. All the algorithms proposed led to an efficient and scalable hardware implementation. The viability of the resulting BbNN system-on-chip (SoC) is proven with real-time performance analysis of real-world case studies, where performance improvements of up to 410$\times$ are observed. The hardware logic utilization is minimized with the help of theoretical analysis and design considerations. A case study requiring the use of recurrent mode BbNN is also presented. All case studies tested with the BbNN give equivalent or better classification accuracies compared to those provided in previous works, but with optimized latency values. As an example, the proposed BbNN solution achieves a classification accuracy of 99.41% for the heart arrhythmia case study, which is an improvement over previous work. The validity of the proposed BbNN model is thus verified.

ABSTRAK

Rangkaian neural boleubah adalah arkitektur yang lebih terkini, dan berbeza daripada rangkaian neural tiruan konvensional (ANN) dalam erti kata bahawa ia membolehkan perubahan dalam struktur dan reka bentuk untuk menghadapi persekitaran yang dinamik. Blok berasaskan rangkaian neural (BbNN) merupakan penyelesaian yang mantap untuk dua masalah asas ANN, iaitu pengoptimuman struktur secara serentak, dan sesuai untuk diimplementasikan di dalam perkakasan bolehubah terbenam seperti *field programmable gate arrays* (FPGA) disebabkan strukturnya yang modular. Walau bagaimanapun, BbNN masih mempunyai beberapa isu-isu tertunggak yang perlu diselesaikan untuk mendapatkan perlaksanaan yang lebih cekap. Reka bentuk perkakasan yang cekap hanya boleh diperolehi dengan pertimbangan reka bentuk yang betul. Sehingga sekarang, tidak ada kerja sebelumnya yang berjaya menonjolkan BbNN yang dikonfigurasikan dalam mod berulang bagi kajian kes yang kompleks, walaupun ia adalah mungkin secara teori. Model BbNN sedia ada tidak jelas menentukan bagaimana latensi sistemnya berfungsi. Ianya juga tidak menentukan bagaimana latensi memberi kesan kepada sistem dan bagaimana ia boleh dioptimumkan. Tambahan pada itu, kaedah semasa yang digunakan untuk melatih BbNN menggunakan algoritma genetik (GA) adalah perlahan, terutamanya untuk set data yang besar. Tesis ini membentangkan model BbNN yang lebih baik, mencadangkan sistem simulasi yang sesuai, dan mengimplementasikannya dalam perkakasan FPGA untuk meningkatkan prestasinya. Ia mempunyai arkitektur yang unggul dengan keluaran yang berketentuan, sekaligus membolehkannya berevolusi dan beroperasi dalam mod suap depan dan berulang. BbNN ini direka untuk mencapai latensi sistem optimum demi memperoleh prestasi yang lebih tinggi, dan membenarkan latihan pada cip untuk pengoptimuman pelbagai objektif dengan penggunaan GA selari. Semua algoritma yang dicadangkan membenarkan implementasi perkakasan yang cekap dan berskala. Kecekapan BbNN di atas sistem-atas-cip terbukti dengan analisis prestasi kajian kes yang kompleks, di mana peningkatan dalam prestasi sehingga 410$\times$ diperhatikan. Penggunaan perkakasan logik dikurangkan dengan bantuan analisis teori dan pertimbangan reka bentuk. Kajian kes yang memerlukan penggunaan BbNN di dalam mod berulang turut dibentangkan. Semua kajian kes yang diuji dengan sistem ini memberikan kadar klasifikasi yang sama atau lebih baik dengan kajian-kajian sebelumnya, tetapi dengan latensi yang optimum. Sebagai contoh, BbNN yang dikemukakan menunjukkan kadar klasifikasi sebanyak 99.41% bagi kes aritmia jantung, yang merupakan peningkatan berbanding dengan kajian sebelumnya. Maka, ini mengesahkan kesahihan model BbNN yang dicadangkan.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ALU	–	Arithmetic Logic Unit
ANN	–	Artificial Neural Network
BPF	–	Band Pass Filter
BbNN	–	Block-based Neural Network
CPU	–	Central Processing Unit
DDR SDRAM	–	Double Data Rate Synchronous Dynamic Random-Access Memory
DPI	–	Direct Programming Interface
DUT	–	Device-Under-Test
DUV	–	Device-Under-Verification
EA	–	Evolutionary Algorithm
ECG	–	Electrocardiogram
FFT	–	Fast Fourier Transform
FPU	–	Floating Point Unit
FPGA	–	Field Programmable Gate Array
FSM	–	Finite State Machine
GA	–	Genetic Algorithm
GCC	–	GNU Compiler Collection
GNU	–	Gnu's Not Unix
GUI	–	Graphical User Interface
HDL	–	Hardware Description Language
HF	–	High Frequency
HPF	–	High Pass Filter
HRV	–	Heart Rate Variability
HW	–	Hardware
HW-SW	–	Hardware-Software
IC	–	Integrated Circuit
I/O	–	Input/Output

LE	–	Logic Elements
LF	–	Low Frequency
LPF	–	Low Pass Filter
LUT	–	Lookup Table
Mbit	–	Mega Bits
MDIPS	–	Millions of Dhrystone Instructions Per Second
MHz	–	Mega Hertz
MLP	–	Multilayer perceptron
MMU	–	Memory Management Unit
ms	–	Millisecond
MUX	–	Multiplexer
MWIPS	–	Millions of Whetstone Instructions Per Second
NoC	–	Network-on-Chip
ns	–	Nanosecond
OS	–	Operating System
PC	–	Personal Computer
PWL	–	Piecewice Linear
PWQ	–	Piecewice Quadratic
RTL	–	Register Transfer Level
RTOS	–	Real Time Operating System
SoC	–	System-on-Chip
SOM	–	Self Organizing Maps
SW	–	Software
USB	–	Universal Serial Bus
VHDL	–	Very High Speed Integrated Circuit Hardware Description Language
VLF	–	Very Low Frequency
WFDB	–	Waveform Database

LIST OF SYMBOLS

x_{0-3}	–	Inputs of a neuron block
y_{0-3}	–	Outputs of a neuron block
w_{0-5}	–	Weight values for the synapses inside a neuron block
b_{0-3}	–	Bias values for the nodes inside a neuron block
d_{0-3}	–	Direction values for the neuron block interconnects
r	–	The amount or rows in a BbNN structure
c	–	The amount or columns in a BbNN structure
f_{BbNN}	–	Fitness value of the BbNN structure
P_{err}	–	BbNN error parameter
P_{LQ}	–	BbNN latency quality parameter
P_{pen}	–	BbNN penalty parameter
α	–	General scaling or modifier constant
β	–	General scaling or modifier constant
β_{1-3}	–	BbNN parameter scaling constants
L_s	–	Specified latency value of the BbNN system
L_{min}	–	Minimum possible latency value
L_{max}	–	Maximum possible latency value
σ	–	Latency modifier for L_{max}
ϕ	–	Latency offset for L_{min}
N_{inv}	–	Number of neuron blocks with invalid configurations
$g(x)$	–	General representation of an activation function
s_p	–	Activation function saturation point
C_a	–	Classification accuracy
T_P	–	The number of true positives
F_P	–	The number of false positives
T_N	–	The number of true negatives
F_N	–	The number of false negatives

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

An artificial neural network (ANN) is a mathematical model inspired by biological neural networks [1]. A neural network consists of an interconnected group of artificial neurons, and it processes information using a connectionist approach to computation. Typically, ANNs are used in adaptive systems that require a training phase to properly configure itself to perform a classification or control task. Figure 1.1 depicts a multilayer perceptron ANN with a single hidden layer.

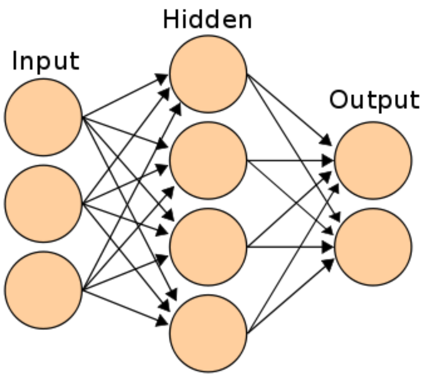

Figure 1.1: A multilayer-perceptron ANN with a single hidden layer.

ANNs have been successfully deployed in solving various kinds of classification and control problems, which include speech recognition, image analysis, adaptive control, and biomedical signal analysis [1–3]. Typically, existing ANN solutions are based on statistical estimations of a given complex problem, in which the relation between the datasets are inferred through heuristics during the training process.

However, ANNs have been criticized for many shortcomings [4, 5]. One weakness of conventional ANNs is the difficulty of obtaining an optimal structure for a given problem. Another issue is that the internal structure of conventional ANNs are rigid, massively interconnected, and allows for little change, making it unsuitable for digital hardware implementation. Also, most types of neural networks are computationally intensive and cannot be implemented in embedded software without sacrificing real time execution [5].

1.1 Block-based Neural Networks

Evolvable neural networks are a more recent architecture, and differs from the conventional ANNs in the sense that it allows changes in the structure and design to cope with dynamic operating environments [6]. Block-based neural networks (BbNNs) were first introduced in 2001 by Moon and Kong [7]. It provides a more unified solution to the two fundamental problems of ANN, which include simultaneous optimization of structure, and viable implementation in reconfigurable digital hardware such as field programmable gate arrays (FPGAs). A BbNN structure is a network of neuron blocks interconnected in the form of a grid as shown in Figure 1.2. A neuron block is a basic data processing unit comprising of a neural network having four variable input/output nodes.

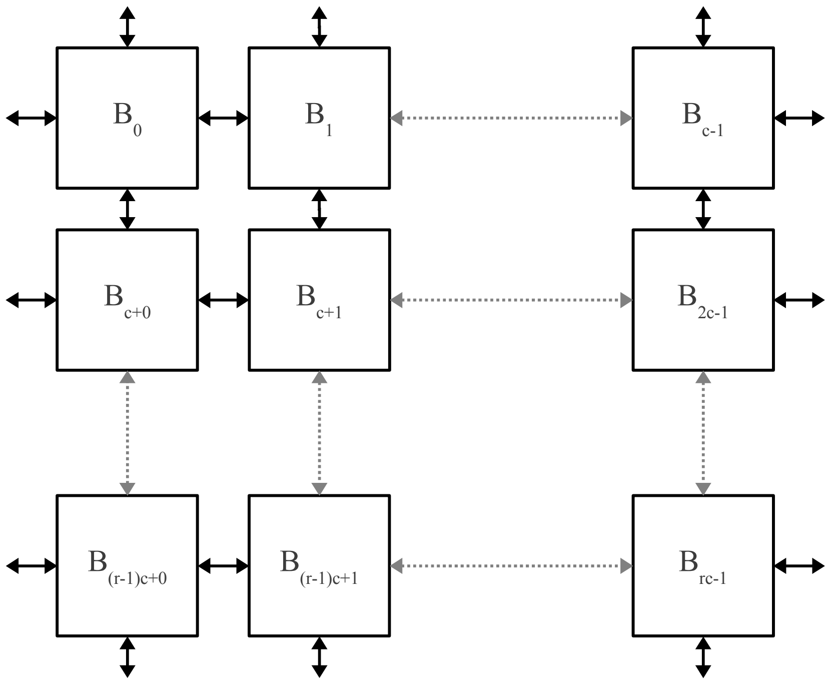

Figure 1.2: Neuron blocks interconnected as a scalable grid to form a BbNN structure [8].

BbNNs have a modular structure, allowing it to be scaled easily by adding or removing neuron blocks. It has been successfully deployed for various kinds of classification problems [6], and is specifically meant to be implemented in digital hardware due to its regular structure. The number of rows (r) and columns (c) differ according to the complexity of the problem being tackled. In addition to this, the internal configuration of the neuron blocks can also change or evolve according to the problem being tackled. These two structural aspects can change dynamically, and provide the evolutionary feature of a BbNN system.

Optimization algorithms are used for training BbNN structures, as is in the case of most evolvable ANNs. Almost all previous work reported in literature make use of genetic algorithm (GA), a search heuristic that mimics the process of natural evolution [9]. The GA methodology has been shown to be able to solve difficult and complex problems that fall in the domain of optimization and search. In GA, a chromosome represents a possible solution, and the chromosome is divided into genes which represent parameters of the solution to a problem. Hence in digital systems, a chromosome is represented in the form of a binary string [10]. In the case of BbNN training, the chromosome represents the synaptic weights and configuration of the BbNN structure.

BbNNs have been successfully used in various applications such as ECG signal classification [8], hypoglycemia detection [11], pattern recognition [12], heart rate variation [13, 14], network intrusion [15, 16], mobile robot control [7], dynamic fuzzy control [17] and many more [6]. Most of these works report very high classification rates, often outperforming equivalent regular ANNs designs. This is mostly due to its evolvable structure.

1.2 Problem Statement

The main advantage of BbNNs when compared to other forms of evolvable ANNs is its modular structure, making it a very promising candidate for an efficient hardware implementation. Hardware implementation platforms such as FPGAs have limited logic elements, and efficient neural network designs allow better resource utilization. Efficient hardware implementation of BbNN structures is the primary goal of this thesis.

Even though BbNNs give relatively promising results as reported in current literature, there are still several outstanding issues to be resolved with currently defined models. These relate to the modeling, hardware design, and optimization of BbNN structures. This thesis tackles these issues by redefining certain aspects of BbNN operation in order to obtain an improved model for better hardware and software implementation. It is likely that the reluctant adoption of BbNNs in practical applications is related to these outstanding issues. Also, to properly apply BbNNs in complex real world problems, it should ideally be implemented in hardware, and must be able to handle on-chip training.

Despite the wide range of applications reported in literature, to date there has been no work reported on BbNNs configured in recurrent mode for complex case studies, even though it is theoretically possible [7]. This is due to a lack of a time-step model required for the design and simulation of recurrent neural networks. Figure 1.3 shows an example of a 2×2 BbNN structure configured in recurrent mode. The behavior of such a configuration in hardware would be indeterministic if a registered architecture is not used, and would normally result in an asynchronous feedback that can lead to circuit metastability. Existing BbNN models in literature do not discuss registering the outputs of the neuron blocks in depth, nor do they include a latency variable that is essential for properly using such models.

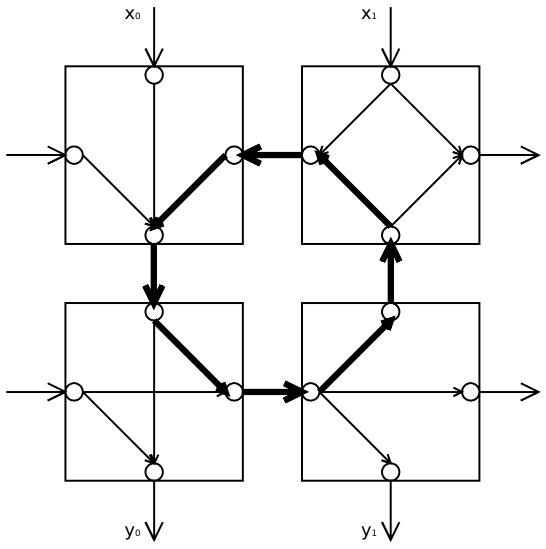

Figure 1.3: Example of a 2×2 BbNN structure configured in recurrent mode.

Related to the recurrent BbNN issue mentioned above, existing BbNN models do not explicitly specify or model the latency of the system, determine how it affects the system, nor how it can be optimized. This is important because optimizing the latency not only provides obvious benefits such as improved performance and reduced power consumption, but also allows the BbNN to behave in a more deterministic manner. This deterministic behavior will allow BbNNs to be able to function properly in recurrent or feedback mode, and can even have internal configurations that would have been normally deemed invalid. Also, previous works on BbNNs [6, 11, 12, 15, 17–20] are difficult to be repeated without latency control because specifying different latency values will often cause different results to appear at the outputs. This thesis proposes a redefined BbNN model with latency control taken into account.

Another issue with BbNN structures is in regards to its hardware design. Previous work only presented the modeling of the BbNN, after which a hardware design is obtained [20, 21]. The methodology and process of mapping the models to hardware are not provided, thus leaving several open questions on the design considerations such as the architectural options, system partitioning, selected activation function, lower/upper limits of internal network parameters (synaptic weight and neuron biases), number representation for arithmetic operations (fixed or floating point), bit precision, finite state machine flow, and register transfer level (RTL) design.

The optimum range for the lower/upper limits of internal network parameters, such as the synaptic weight, neuron biases, and activation function saturation points are not clearly defined in existing work. An efficient hardware design can only be obtained after the optimum range of these values is known. The main rationale of using BbNNs instead of conventional ANNs is the advantage in hardware utilization, but this cannot be achieved if the hardware design is not efficient. This thesis explores this aspect of BbNN modeling with the goal of obtaining an optimal neuron block design for effective hardware implementation.

Current methods of training BbNNs using GA are slow, especially with large training datasets. The problem worsens when training for recurrent BbNNs whereby the convergence rate will drop significantly. An improved GA mechanism is required to improve speed and convergence rate, as well as to allow multithreaded simulation of BbNN structures for parallel fitness evaluation. This will allow the use of superior computational techniques to be applied on BbNN training, such as supercomputing clusters or parallelized FPGAs.

BbNNs, like most evolvable ANNs, have complex neuron interconnects. The interconnections are simple for small BbNN structures, but gradually increase in complexity as the structure grows bigger. Thus a parameterizable interconnection algorithm is required for experimentation, and will not only be useful for creating scalable BbNN structures in simulation models, but also facilitate automated generation of interconnects in hardware implementations. To date, no such algorithm exists in published literature. Such an algorithm would be complex and prone to errors, and the lack of this in current literature is possibly another reason for the reluctant adoption of BbNN systems.

1.3 Objectives

The primary objective of this thesis is to improve on existing BbNN models, to propose a state-of-the-art simulation and co-design environment for it, and to implement it on hardware for performance analysis. In detail, the objectives of this thesis are:

1. To propose a block-based neural network (BbNN) architecture that has the following novel features:
    - Evolvable architecture with deterministic outputs that can operate in both feedforward and recurrent mode.
    - Redesigned for optimal system latency to achieve higher performance.
    - On-chip training and multi-objective optimization using state-of-the-art parallel genetic algorithms.
    - Allows for an efficient and scalable hardware implementation.
    - Provides a platform for effective solutions to real world complex classification problems.
2. To develop an effective implementation platform for practical BbNN solutions that is based on FPGA embedded hardware, and prove the viability of the resulting hardware-software design with real-time performance analysis of complex, real world case studies.

1.4 Scope of Work

The work in this thesis uses a combination of tools mostly obtained from open source software and libraries. This allows for a work that is easier to repeat in the future. The platform for testing and using BbNN models are built from scratch using a variety of tools. The software tools and prototyping platform applied in this thesis are described as follows:

a) Algorithmic models are verified and analyzed using GNU Octave, an open source Matlab alternative freely available on Linux. It is also used for graph plotting, data preprocessing, and equation optimization.

b) The proposed BbNN software model is developed in C/C++ and modeled using the proposed array interconnect algorithm. It is compiled with the GCC compiler under Ubuntu Linux, with all compiler optimizations turned on for maximum performance (optimization level 3).

c) This work utilizes GAlib, a powerful C/C++ open source library for applying GA optimization in software [22]. The optimization algorithm applied is scoped to a multiple population version of steady state GA, which is better suited for multi-objective problems.

d) Parallel processing for BbNN simulation and training is achieved using a computer equipped with a 2.8 Ghz Intel Core i7 processor. For performing multithreaded execution, the *libpthread* library is used.

e) Verilog and SystemVerilog HDL is used to model the hardware design of the BbNN. Hardware prototyping and implementation of the BbNN system as an SoC is done on an Altera Stratix III FPGA as shown in Fig. 1.4. The Nios II processor used is clocked at 266 MHz, and provides a platform in which the Nios2-Linux operating system can execute. This allows the use of the GAlib library, filesystem support, and USB host support (through *libUSB*) for embedded systems. It also allows the usage of external USB-based sensors.

f) The neuron blocks are verified in Icarus Verilog and Modelsim using SystemVerilog tesbenches. The DPI-C interface in Modelsim is used for more advanced testbenches to incorporate co-verification with C/C++ golden reference models. The BbNN structure is constructed by combining these neuron blocks to form an array.

g) The BbNN structure is modeled in such a way that the rows and columns are parameterizable, by using the proposed array interconnect algorithm in conjunction with the Verilog *generate* command. SystemVerilog is used for the top-level BbNN structure because of the better support for multi-dimensional arrays for signals and registers.

h) The entire BbNN hardware structure is verified using co-simulation with C/C++ due to the complexity of the design. Verilator is used for this purpose.

i) The case studies used to verify and analyze the performance of the proposed BbNN model are limited to the following problems:

   i The XOR classification problem.

   ii Driver drowsiness detection based on HRV.

   iii Heart arrhythmia classification from ECG signals.

   iv Grammar inference of the Tomita language.

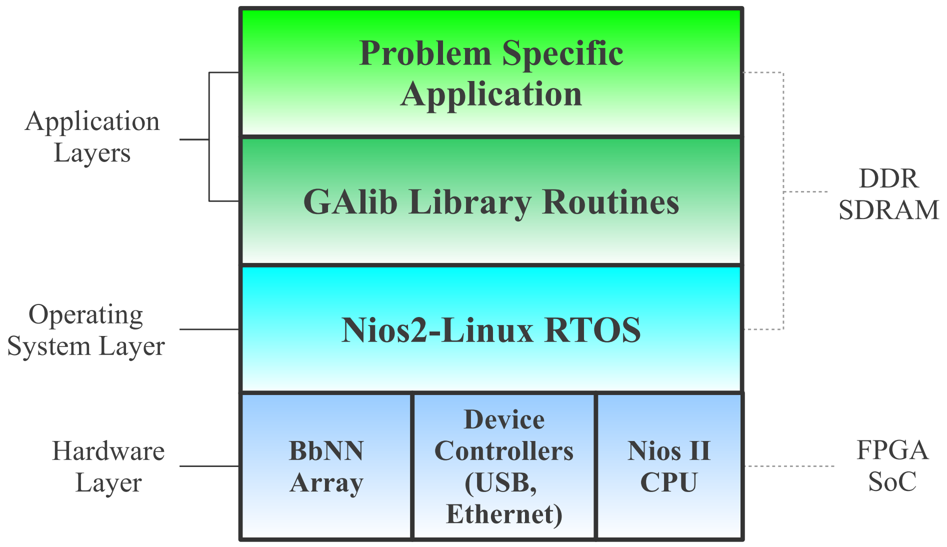

Figure 1.4: Layers of the proposed embedded BbNN SoC.

1.5 Contributions

The proposed BbNN model in this thesis has an improved architecture over existing work. A platform is developed for the simulation, verification, and implementation of the proposed BbNN model. In summary, the main contributions of this thesis are:

- Based on all known previous works, this work is among the first successful attempts at training a BbNN in recurrent mode, hence demonstrating the ability to evolve recurrent structures through feedback signals with the help of latency optimization. The recurrent mode BbNN is applied in a test case study on grammar inference using randomly generated datasets from the Tomita language [23–25].

- This thesis is also among the first work in the field to describe and optimize the latency of evolvable BbNNs using multi-objective genetic algorithm (GA) during the training process. This is done by introducing the latency value as a gene in the BbNN chromosome, and is simultaneously optimized during the training process. A suitable fitness function for the GA is proposed for achieving this.

- This work proposes an effective hardware implementation platform for BbNN with on-chip training. The performance of the hardware implementation allows real-time classification.
- A novel multithreaded solution is proposed for accelerating the training of BbNN simulation models, which are repetitively evaluated by the GA fitness function. Significant reduction in simulation time is obtained when compared to using a only a single thread.
- An equation describing the optimum range for the upper/lower boundaries of the internal BbNN parameters is proposed. This equation was obtained through geometric analysis, and is empirically proven. Using this equation, it is possible to select an optimum range for these parameters that is suitable for resource constrained hardware implementation platforms.
- This thesis also presents the RTL design of a neuron block with properly described methodology, using a single multiplier for each neuron block. A novel method of reusing the multiplier to smoothly approximate a hyperbolic tangent (*tanh*) function to be used as the activation function for the neuron blocks is presented. This is an important contribution, because a sigmoid-like activation function (which provides faster learning rates [26,27]) is provided at almost zero cost.
- This thesis also provides a parameterizable BbNN array interconnect algorithm that works for software simulation models as well as hardware implementation models. Without this algorithm, designing a BbNN array will pose an obstacle due to the complexity of the neuron block interconnects, especially when the size of the array needs to be changed during the experiments.

1.6 Dissertation Organization

The thesis is organized as follows.

Chapter 2 covers literature review and discusses previous work done, including works on the case studies used in this thesis. It also covers all related background theory.

Chapter 3 covers the methodology for the work done in this thesis. This also includes the general approach taken for the research done in this work, as well as the tools and platform used. It also includes a section describing the case studies used and the methodology for creating the datasets.

Chapter 4 presents the development of the models, algorithms, system latency and design considerations for the proposed BbNN model. The architecture of a parameterizable multithreaded simulation model is discussed. The application of GA for performing multi-objective optimization in the training of the proposed BbNN is also presented.

Chapter 5 provides a complete description on the RTL design used for the implementation of the proposed BbNN system in hardware, and an overview of the hardware system architecture. Hardware related design considerations are discussed here.

Chapter 6 presents the results and analysis of all the experimentation done in this thesis, including the geometric analysis for the synaptic weight boundaries of the neuron blocks, the results of the four different case studies used, the hardware verification results, and the performance analysis of all proposed BbNN models.

Chapter 7 summarizes the thesis, re-stating the contributions, and suggest directions for future research.

REFERENCES

1. Haykin, S. S. *Neural networks and learning machines*. vol. 3. Prentice Hall. 2009.
2. Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer. 2006.
3. Hassoun, M. *Fundamentals of artificial neural networks*. MIT press. 1995.
4. Francis, L. Neural networks demystified. *Casualty Actuarial Society Forum*. 2001. 253–320.
5. Skrbek, M. Fast neural network implementation. *Neural Network World*, 1999. 9(5): 375–391.
6. Merchant, S. G. and Peterson, G. D. Evolvable block-based neural network design for applications in dynamic environments. *VLSI Design*, 2010. 2010: 25. ISSN 1065-514X.
7. Moon, S. W. and Kong, S. G. Block-based neural networks. *IEEE Transactions on Neural Networks*, 2001. 12(2): 307–17. ISSN 1045-9227.
8. Jiang, W. and Kong, S. G. Block-based neural networks for personalized ECG signal classification. *IEEE Transactions on Neural Networks*, 2007. 18(6): 1750–61. ISSN 1045-9227.
9. Holland, J. H. *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: University of Michigan Press. 1975.
10. Goldberg, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Upper Saddle River,NJ, USA: Addison-Wesley Professional. 1989.
11. San, P. P., Ling, S. H. and Nguyen, H. T. Block based neural network for hypoglycemia detection. *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE. 2011. 5666–5669.
12. Moon, S. and Kong, S. Pattern recognition with block-based neural networks. *Proceedings of the 2002 International Joint Conference on Neural Networks, 2002 (IJCNN'02)*. 2002, vol. 1. ISBN 0780372786. 992–996.

13. Nambiar, V. P., Khalil-Hani, M., Sia, C. and Marsono, M. Evolvable Block-based Neural Networks for classification of driver drowsiness based on heart rate variability. *2012 IEEE International Conference on Circuits and Systems (ICCAS)*. 2012. 156–161.

14. Khalil-Hani, M., Sia, C. W., Shaikh-Husin, N. and Nambiar, V. P. 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.

15. Tran, Q. A., Jiang, F. and Ha, Q. M. Evolving Block-Based Neural Network and Field Programmable Gate Arrays for Host-Based Intrusion Detection System. *2012 Fourth International Conference on Knowledge and Systems Engineering (KSE)*. IEEE. 2012. 86–92.

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

17. Karaköse, M. and Akin, E. *Dynamical Fuzzy Control with Block Based Neural Network*. Technical report. Technical Report, Department of Computer Engineering, Firat University. 2006.

18. Jiang, W. and Kong, S. A Least-Squares Learning for Block-based Neural Networks. *Advances in Neural Networks, A Supplement (DCDIS)*, 2007. 14(SI): 242–247.

19. Haridass, S. and Hoe, D. Fault Tolerant Block Based Neural Networks. *System Theory (SSST), 2010 42nd Southeastern Symposium on*. IEEE. 2010. ISBN 9781424456925. 357–361.

20. Jewajinda, Y. and Chongstitvatana, P. A parallel genetic algorithm for adaptive hardware and its application to ECG signal classification. *Neural Computing and Applications*, 2012: 1–18.

21. Kothandaraman, S. *Implementation of block-based neural networks on reconfigurable computing platforms*. Master's Thesis. The University of Tennessee, Knoxville. 2004.

22. Wall, M. *GAlib : A C ++ Library of Genetic Algorithm Components*. Technical Report August. Massachusetts Institute of Technology. 1996.

23. Blanco, A., Delgado, M. and Pegalajar, M. A genetic algorithm to obtain the optimal recurrent neural network. *International Journal of Approximate*

Reason, 2000. 23(July 1999): 67–83.

24. Giles, C., Chen, D., Miller, C., Chen, H., Sun, G. and Lee, Y. Second-order recurrent neural networks for grammatical inference. *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*. Ieee. 1991. ISBN 0-7803-0164-1. 273–281.

25. Angeline, P., Saunders, G. and Pollack, J. An Evolutionary Algorithm that Constructs Recurrent Neural Networks An Evolutionary Algorithm that Constructs Recurrent Neural Networks. *Neural Networks, IEEE Transactions on*, 1994. 5(1): 54–65.

26. Omondi, A. and Rajapakse, J. *FPGA implementations of neural networks*. vol. 1. Springer New York, NY, USA:. 2006.

27. Jordan, M. *et al.* Why the logistic function? A tutorial discussion on probabilities and neural networks, 1995.

28. Cheung, V. and Cannons, K. *An Introduction to Neural Networks*. Technical report. Signal & Data Compression Laboratory, Electrical & Computer Engineering University of Manitoba, Winnipeg, Manitoba, Canada. 2002.

29. McCullagh, P. and Nelder, J. A. *Generalized Linear Models, Second Edition (Chapman & Hall/CRC Monographs on Statistics & Applied Probability)*. Chapman and Hall. 1989.

30. Teuscher, C. FPGA Implementations of Neural Networks (Ormondi. A.R. and Rajapakse, J.C., Eds.; 2006). *IEEE Transactions on Neural Networks*, 2007. 18(5): 1550. ISSN 1045-9227. doi:10.1109/TNN.2007.906886.

31. Kwan, H. Simple sigmoid-like activation function suitable for digital hardware implementation. *IET Electronics Letters*, 1992. 28(15): 1379–1380.

32. Moreno, F., Alarcon, J., Salvador, R. and Riesgo, T. FPGA implementation of an image recognition system based on Tiny Neural networks and on-line reconfiguration. *Industrial Electronics, 2008. IECON 2008. 34th Annual Conference of IEEE*. IEEE. 2008. 2445–2452.

33. Kong, S. G. Time series prediction with evolvable block-based neural networks. *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*. IEEE. 2004, vol. 2. 1579–1583.

34. Jiang, W., Kong, S. and Peterson, G. ECG signal classification using block-based neural networks. *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*. IEEE. 2005, vol. 1. 326–331.

35. Merchant, S., Peterson, G. D., Park, S. K. and Kong, S. G. FPGA

Implementation of Evolvable Block-based Neural Networks. *IEEE Congress on Evolutionary Computation, 2006 (CEC 2006)*. 2006. 3129–3136.

36. Jewajinda, Y. An adaptive hardware classifier in FPGA based-on a cellular compact genetic algorithm and block-based neural network. *2008 International Symposium on Communications and Information Technologies (ISCIT 2008)*. 2008. ISBN 9781424423361. 658–663.

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

38. Samarah, A., Habibi, A., Tahar, S. and Kharma, N. Automated Coverage Directed Test Generation Using a Cell-Based Genetic Algorithm. *High-Level Design Validation and Test Workshop, 2006. Eleventh Annual IEEE International*. 2006. ISSN 1552-6674. 19–26. doi:10.1109/HLDVT.2006.319996.

39. Sivanandam, S. N. and Deepa, S. N. *Introduction to genetic algorithms*. Springer. 2008.

40. Weise, T. *Global Optimization Algorithms {–} Theory and Application*. it-weise.de (self-published): {Germany}. 2009. URL `http://www.it-weise.de/projects/book.pdf`.

41. Nickray, M., Dehyadgari, M. and Afzali-kusha, A. Power and delay optimization for network on chip. *Circuit Theory and Design, 2005. Proceedings of the 2005 European Conference on*. IEEE. 2005, vol. 3. III–273.

42. Bhardwaj, K. and Jena, R. Energy and bandwidth aware mapping of IPs onto regular NoC architectures using multi-objective genetic algorithms. *System-on-Chip, 2009. SOC 2009. International Symposium on*. 2009. 27–31.

43. Ascia, G., Catania, V. and Palesi, M. Multi-objective mapping for mesh-based NoC architectures. *Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*. 2004. ISBN 1581139373. 182–187.

44. Morgan, A. a., Elmiligi, H., El-Kharashi, M. W. and Gebali, F. Multi-objective optimization for Networks-on-Chip architectures using Genetic Algorithms. *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*. Ieee. 2010, 1. ISBN 978-1-4244-5308-5. 3725–3728.

45. Rogado, E., Garcia, J. L., Barea, R., Bergasa, L. M. and Lopez, E. Driver

fatigue detection system. *Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on*. IEEE. 2009. 1105–1110.

46. Michail, E., Kokonozi, A., Chouvarda, I. and Maglaveras, N. EEG and HRV markers of sleepiness and loss of control during car driving. *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*. IEEE. 2008, vol. 2008. ISBN 9781424418152. ISSN 1557-170X. 2566–2569. doi:10.1109/IEMBS.2008.4649724.

47. Furman, G. D., Baharav, A. and Cahan, C. Early detection of falling asleep at the wheel: A heart rate variability approach. *in Cardiology, 2008*. 2008. 1109–1112. doi:10.1109/CIC.2008.4749240.

48. Jeong, I. C., Lee, D. H., Park, S. W., Ko, J. I. and Yoon, H. R. Automobile driver's stress index provision system that utilizes electrocardiogram. *Intelligent Vehicles Symposium, 2007 IEEE*. IEEE. 2007. 652–656.

49. Bonnet, M. H. and Arand, D. L. Heart rate variability: sleep stage, time of night, and arousal influences. *Electroencephalography and Clinical Neurophysiology*, 1997. 102(5): 390–396. ISSN 0013-4694.

50. Ramesh, M. V., Nair, A. K. and Kunnathu, A. T. Real-Time Automated Multiplexed Sensor System for Driver Drowsiness Detection. *Wireless Communications, Networking and Mobile Computing (WiCOM), 2011 7th International Conference on*. IEEE. 2011. 1–4.

51. McCartt, A. T., Rohrbaugh, J. W., Hammer, M. C. and Fuller, S. Z. Factors associated with falling asleep at the wheel among long-distance truck drivers. *Accident Analysis & Prevention*, 2000. 32(4): 493–504.

52. Gislason, T., Tomasson, K., Reynisdottir, H., Björnsson, J. K. and Kristbjarnarson, H. Medical risk factors amongst drivers in single-car accidents. *Journal of internal medicine*, 1997. 241(3): 217–223.

53. Podrid, P. and Kowey, P. *Cardiac arrhythmia: mechanisms, diagnosis, and management*. Lippincott Williams & Wilkins. 2001.

54. Wolf, P., Dawber, T., Thomas Jr, H. and Kannel, W. Epidemiologic assessment of chronic atrial fibrillation and risk of stroke. *Neurology*, 1978. 28(10): 973–973.

55. Moody, G. and Mark, R. The MIT-BIH arrhythmia database on CD-ROM and software for use with it. *Computers in Cardiology 1990, Proceedings*. IEEE. 1990. 185–188.

56. Tsai, Y., Hung, B. and Tung, S. An experiment on ECG classification

using back-propagation neural network. *Engineering in Medicine and Biology Society, 1990., Proceedings of the Twelfth Annual International Conference of the IEEE*. IEEE. 1990. 1463–1464.

57. Prasad, G. and Sahambi, J. Classification of ECG arrhythmias using multi-resolution analysis and neural networks. *TENCON 2003. Conference on Convergent Technologies for Asia-Pacific Region*. IEEE. 2003, vol. 1. 227–231.

58. Lagerholm, M., Peterson, C., Braccini, G., Edenbrandt, L. and Sornmo, L. Clustering ECG complexes using Hermite functions and self-organizing maps. *Biomedical Engineering, IEEE Transactions on*, 2000. 47(7): 838–848.

59. Nambiar, V. P., Khalil-Hani, M. and Marsono, M. Evolvable Block-Based Neural Networks for Real-Time Classification of Heart Arrhythmia From ECG Signals. *2012 IEEE EMBS Conference on Biomedical Engineering & Sciences*. 2012.

60. Watrous, R. L. and Kuhn, G. M. Induction of finite-state languages using second-order recurrent networks. *Neural Computation*, 1992. 4(3): 406–414.

61. Delgado, M. and Pegalajar, M. A multiobjective genetic algorithm for obtaining the optimal size of a recurrent neural network for grammatical inference. *Pattern Recognition*, 2005. 38(9): 1444–1456. ISSN 00313203. doi:10.1016/j.patcog.2004.03.026.

62. Cochran, J., Horng, S. and Fowler, J. A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines. *Computers & Operations Research*, 2003. 30(7): 1087–1102.

63. Altera. Using ModelSim to Simulate Logic Circuits for Altera FPGA Devices. Published online, 2011.

64. Mentor Graphics. Modelsim Users Manual. Published online, 2011.

65. Williams, S. Homepage for the Icarus Verilog project. Published online, 2012. URL `http://iverilog.icarus.com`.

66. Snyder, W., Galbi, D. and Wasson, P. Free Verilog and SystemC Software - Serious Tools for Real Projects. Published online, 2012. URL `http://www.veripool.org/wiki/verilator`.

67. Bennett, J. Processor Verification using Open Source Tools and the GCC Regression Test Suite: A Case Study. *Design Verification Club Meeting*. Infineon, Bristol. 2010. 1–13.

68. Ungerer, G., Dionne, J. and Durant, M. uClinux: Embedded

Linux/Microcontroller Project. Published online, 2008. URL `http://www.uclinux.org`.

69. Spear, C. *SystemVerilog for Verification: A Guide to Learning the Testbench Language Features*. Springer Verlag. 2008.

70. Tomita, M. *Efficient parsing for natural language: a fast algorithm for practical systems*. vol. 8. Springer. 1985.

71. De Jong, K. and Spears, W. Using genetic algorithms to solve NP-complete problems. *Proceedings of the third international conference on Genetic algorithms*. Morgan Kaufmann, San Mateo, CA. 1989, vol. 124. 132.

72. Yates, R. Fixed-point arithmetic: An introduction. *Digital Signal Labs*, 2009. 81(83): 198.

73. Khalil-Hani, M. *Digital Systems: VHDL & Verilog Design*. UTM Skudai: Prentice Hall. 2012.

74. Cummings, C. The fundamentals of efficient synthesizable finite state machine design using nc-verilog and buildgates. *Proceedings of International Cadence Usergroup Conference*. 2002. 16–18.

75. Merchant, S. G. and Peterson, G. D. An evolvable artificial neural network platform for dynamic environments. *Circuits and Systems, 2008. MWSCAS 2008. 51st Midwest Symposium on*. 2008. ISSN 1548-3746. 77–80.

76. Longbottom, R. Roy Longbottom's PC Benchmark Collection. Published online, 2012. URL `http://www.roylongbottom.org.uk`.

APPENDIX A

PUBLICATIONS

This appendix shows the papers written based on the results obtained from the work done in this thesis. It also includes papers that are related to the work done in thesis, such as SoC design, GA-based hardware designs, embedded hardware and computing systems. The following is a summary of these papers:

1. Vishnu P. Nambiar, Khalil-Hani, M., & Marsono, M. N. (2012). HW/SW co-design of reconfigurable hardware-based genetic algorithm in FPGAs applicable to a variety of problems. Springer Computing. (Scopus indexed, IF 0.9). *Published.*

2. Vishnu P. Nambiar, Khalil-Hani, M., & Zabidi, M. M. (2009). Accelerating the AES encryption function in OpenSSL for embedded systems. International Journal of Information and Communication Technology, 2(1/2), 83. (Scopus indexed). *Published.*

3. Vishnu P. Nambiar, Khalil-Hani, M., & Marsono, M. N. (2012). Optimization of Structure and System Latency in Evolvable Block-Based Neural Networks using Genetic Algorithm. IEEE Transactions on Evolutionary Computation. (Scopus indexed, IF 3.3). *Under Review.*

4. Vishnu P. Nambiar, Khalil-Hani, M., & Marsono, M. N. (2013). Hardware Implementation of Evolvable Block-Based Neural Networks Utilizing a Cost Efficient Sigmoid-Like Activation Function. Elsevier Neurocomputing. (Scopus indexed, ISI IF 1.6). *Under Review.*

5. Khalil-Hani, Vishnu P. Nambiar, M., & Marsono, M. N. (2013). Co-Simulation Methodology for Improved Design and Verification of Hardware Neural Networks. Annual Conference of the IEEE Industrial Electronics Society (IECON2013). (Scopus indexed). *Under Review.*

6. Vishnu P. Nambiar, Khalil-Hani, M., & Marsono, M. N. (2012). Evolvable Block-Based Neural Networks for Real-Time Classification of Heart Arrhythmia

From ECG Signals. 2012 IEEE EMBS Conference on Biomedical Engineering & Sciences. (Scopus indexed). *Published.*

7. Vishnu P. Nambiar, Khalil-Hani, M., Sia, C. W., & Marsono, M. N. (2012). Evolvable Block-Based Neural Networks for Classification of Driver Drowsiness based on Heart Rate Variability. IEEE International Conference on Circuits & Systems (ICCAS2012). (Scopus indexed). *Published.*

8. Khalil-Hani, M., Sia, C.W. & Vishnu P. Nambiar. (2012). FPGA-based Embedded System for the detection of Driver Drowsiness using ECG signals. International Conference on Electrical Engineering and Computer Sciences (ICEECS 2012). (Scopus indexed). *Published.*

9. Khalil-Hani, M., Vishnu P. Nambiar, & Marsono, M. (2012). GA-based Parameter Tuning in Finger-Vein Biometric Embedded Systems for Information Security. 2012 1st IEEE International Conference on Communications in China (ICCC) (pp. 236–241). (Scopus indexed). *Published.*

10. Irwansyah, A., Vishnu P. Nambiar, & Khalil-Hani, M. (2009). An AES Tightly Coupled Hardware Accelerator in an FPGA-based Embedded Processor Core. 2009 International Conference on Computer Engineering and Technology (pp. 521–525). (Scopus indexed). *Published.*

11. Khalil-Hani, M., Vishnu P. Nambiar, & Marsono, M. (2010). Hardware Acceleration of OpenSSL cryptographic functions for high-performance Internet Security. Intelligent Systems, Modelling and Simulation (ISMS), 2010 International Conference on (pp. 374–379). IEEE. (Scopus Indexed). *Published.*