University of Cyprus

Department of Electrical and Computer Engineering

Intelligent Monitoring of Hardware Components Using Artificial Neural Networks: A Case study Using Networks on Chip

Andreas Savva

A Dissertation Submitted to the University of Cyprus in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

June, 2024

VALIDATION PAGE

Doctoral Candidate: Andreas Savva

Doctoral Thesis Title: Intelligent Monitoring of Hardware Components Using Artificial Neural Networks: A Case study Using Networks on Chip

The present Doctorate Dissertation was submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in the ***Department of Electrical and Computer Engineering****, and was approved on March 5, 2024 by the members of the* ***Examination Committee****.*

Committee Chair	Dr. Georgios Ellinas, Professor
Research Supervisor	Dr. Theocharis Theocharides, Associate Professor
Research Supervisor	Dr. Chrysostomos Nicopoulos, Associate Professor
Committee Member	Dr. Maria Michael, Associate Professor
Committee Member	Dr. Panayiotis Kolios, Assistant Professor
Committee Member	Dr. Panayiota Nikolaou, Lecturer in Computer Engineering

DECLARATION OF DOCTORAL CANDIDATE

The present doctoral dissertation was submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy of the University of Cyprus. It is a product of original work of my own, unless otherwise mentioned through references, notes, or any other statements.

. . . .

Andreas Savva

Abstract

The aim of this dissertation is to design and develop an intelligent way to monitor hardware with the use of robust Artificial Neural Networks (ANNs). Power consumption, reliability and robustness are major limitations in hardware today and researchers have been constantly working on reducing those issues. ANNs can be used for monitoring and detection purposes since they have excellent characteristics such as generalization capability, robustness and excellent prediction results. For the development of the intelligent monitoring, we will use NoCs (Networks-on-Chips) as case study. NoCs provide scalable on-chip communication and are expected to be the dominant interconnection architectures in multicore and manycore systems.

Several attempts have been made to reduce the power consumption at both the circuit level and the system level. Most past research efforts have proposed selective on/off state switching based on system-level information based on utilization levels. Most of these proposed algorithms focus on a pessimistic and simple static threshold mechanism which determines the turning off/on. This work presents an intelligent dynamic power management policy with improved predictive abilities based on supervised online learning of the system status (i.e. expected future utilization link levels), where links are turned off and on via the use of a small and scalable neural network. Simulation results with various synthetic traffic models over various network topologies show that the proposed work can reach up to 13% power savings when compared to a trivial threshold computation, at very low (< 4%) hardware overheads.

Additionally, this work presents a design exploration framework for developing a high-level ANN for fault detection in hardware systems. Designing an ANN in order to be used for fault detection purposes includes different parameters. Through this work, those parameters are presented and analyzed based on simulations. Moreover, after the development of the ANN, in order to evaluate it, a case study scenario based on Networks on Chip is used for detection of inter router link faults. Simulation results with various synthetic traffic models show that the proposed work can detect up to 99% of inter-router link faults with a delay less than 60 cycles. Added to this, the size of the ANN is kept relatively small and can be implemented in hardware easily. Synthesis results indicate an estimated amount of 0.0633mW power consumption per neuron for the implemented ANN when computing a complete cycle.

Lastly, this work also provides a study for analyzing the robustness of ANNs used for prediction purposes, based on weights alterations and an intelligent method is developed for increasing the robustness of the ANNs with minimum additional hardware overheads. Simulation results with the use of Princeton Application Repository for Shared-Memory Computers (PARSEC) benchmarks show that our method can maintain the robustness of the ANNs at high levels (around 98% prediction accuracy) with minimum additional hardware overheads.

Overall, the contributions of this dissertation strongly advance the state-of-the-art in the field of intelligent monitoring of different hardware and NoCs, combined with the use of innovative intelligent methods and techniques based on ANNs.

Περίληψη

Ο στόχος αυτής της εργασίας είναι να σχεδιάσει και να αναπτύξει ένα έξυπνο τρόπο παρακολούθησης υλικού και δικτύων με τη χρήση εύρωστων Τεχνητών Νευρωνικών Δικτύων (ΤΝΔ). Η κατανάλωση ρεύματος, τα σφάλματα σύνδεσης και η ευρωστία αποτελούν σημαντικους περιορισμούς στο υλικό σήμερα και οι ερευνητές εργάζονται συνεχώς για τη μείωση αυτών των προβλημάτων. Τα ΤΝΔ μπορούν να χρησιμοποιηθούν για σκοπούς παρακολούθησης και ανίχνευσης αφού έχουν εξαιρετικά χαρακτηριστικά όπως ικανότητα γενίκευσης, ευρωστία και εξαιρετικές δυνατότητες πρόβλεψης. Για τη δημιουργία του έξυπνου τρόπου παρακολούθησης, θα χρησιμοποιήσουμε τα Δίκτυα σε Τσιπ (ΔΣΤ) σαν σενάριο μελέτης. Τα ΔΣΤ παρέχουν επεκτάσιμη επικοινωνία στα τσιπ και αναμένεται να είναι οι κυρίαρχες αρχιτεκτονικές δισύνδεσης σε συστήματα πολλαπλών πυρήνων.

Έχουν γίνει αρκετές προσπάθειες για μείωση της κατανάλωσης ισχύος τόσο σε επίπεδο κυκλώματος όσο και σε επίπεδο συστήματος. Οι περισσότερες προηγούμενες ερευνητικές προσπάθειες έχουν προτείνει επιλεκτική εναλλαγή κατάστασης με απενεργοποίηση και ενεργοποίηση με βάση πληροφορίες σε επίπεδο συστήματος με βάση τα επίπεδα χρήσης. Οι περισσότεροι απ' αυτούς τους μηχανισμούς επικεντρώνονται σε έναν απλό μηχανισμό στατικού ορίου το οποίο καθορίζει την απενεργοποίηση/ενεργοποίηση.

Μέσα από τη δουλειά μας παρουσιάζουμε μια έξυπνη και δυναμική πολιτική διαχείρησης ενέργειας με βελτιωμένες προγνωστικές ικανότητες που βασίζεται σε εποπτευόμενη εκμάθηση της κατάστασης του συστήματος (δηλαδή αναμενόμενα μελλοντικά επίπεδα χρήσης συνδέσεων), όπου οι σύνδεσμοι απενεργοποιούνται και ενεργοποιούνται μέσω της χρήσης ενός μικρού και επεκτάσιμου νευρωνικού δικτύου. Τα αποτελέσματα προσομοίωσης μέσω του πλαισίου το οποίο αναπτύξαμε με διάφορα μοντέλα συνθετικής και ρεαλιστικής κυκλοφορίας σε διάφορες τοπολογίες δικτύων δείχνουν ότι ο προτεινόμενος μηχανισμός μπορεί να φτάσει εώς και 13% εξοικονόμηση ενέργειας σε σύγκριση με ένα τετριμμένο μηχανισμό υπολογισμού ορίου, σε πολύ χαμηλά (4%) γενικά έξοδα υλικού.

Επιπλεόν, αυτή η δουλειά παρουσιάζει ένα πλαίσιο εξερεύνησης σχεδιασμού για την ανάπτυξη ενός τεχνητού νευρωνικού δικτύου υψηλού επιπέδου για την ανίχνευση σφαλμάτων σε συστήματα υλικού. Ο σχεδιασμός ενός ΤΝΔ προκειμένου να χρησιμοποιηθεί για σκοπούς ανίχνευσης σφαλμάτων περιλαμβάνει διάφορες παραμέτρους. Μέσα από αυτή την εργασία, αυτές οι παράμετροι παρουσιάζονται και αναλύονται με βάση προσομοιώσεις. Επιπλέον, μετά την ανάπτυξη του ΤΝΔ, για την αξιολόγησή του, χρησιμοποιείται ένα σενάριο μελέτης βασισμένο σε ΔΣΤ για τον εντοπισμό σφαλμάτων σύνδεσης μεταξύ των δρομολογητών. Τα αποτελέσματα προσομοίωσης με διάφορα μοντέλα κυκλοφορίας δείχνουν ότι ο προτεινόμενος μηχανισμός μπορεί να ανιχνεύσει έως και το 99% των σφαλμάτων σύνδεσης μεταξύ των δρομολογητών με καθυστέρηση μικρότερη από 60 κύκλους. Επιπλέον, το μέγεθος του ΤΝΔ διατηρείται σχετικά μικρό και μπορεί να εφαρμοστεί εύκολα σε υλικό. Τα αποτελέσματα της σύνθεσης υποδεικνύουν μια εκτιμώμενη ποσότητα κατανάλωσης ισχύος 0,0633 mW ανά νευρώνα για το υλοποιημένο ΤΝΔ κατά τον υπολογισμό ενώς πλήρους κύκλου.

Τέλος, αυτή η εργασία παρέχει επίσης μια μελέτη για την ανάλυση της ευρωστίας των ΤΝΔ που χρησιμοποιούνται για σκοπούς πρόβλεψης, με βάση τις αλλαγές βαρών και αναπτύσσεται επίσης μια έξυπνη μέθοδος για την αύξηση της ευρωστίας με ελάχιστα πρόσθετα έξοδα υλικού. Τα αποτελέσματα προσομοίωσης με τη χρήση ρεαλιστικών μοντέλων κίνησης με σημεία αναφοράς από το Princeton Application Repository for Shared-Memory Computers (PARSEC) benchmark suite δείχνουν ότι η μέθοδός μας μπορεί να διατηρήσει την ευρωστία των ΤΝΔ σε υψηλά επίπεδα (περίπου 98% ακρίβεια πρόβλεψης) με ελάχιστα πρόσθετα έξοδα υλικού.

Συνολικά, οι συνεισφορές αυτής της διατριβής προάγουν έντονα την τελευταία λέξη της τεχνολογίας στον τομέα της έξυπνης παρακολούθησης υλικού και ΔΣΤ σε συνδυασμό με τη χρήση καινοτόμων έξυπνων μεθόδων και τεχνικών που βασίζονται σε ΤΝΔ.

Acknowledgments

I would like to express my deepest gratitude to everyone who has supported me through the completion of this dissertation. Firstly, I would like to thank my supervisors Dr. Theocharis Theocharides and Dr. Chrysostomos Nicopoulos for their unwavering guidance and support. I would also like to thank the University of Cyprus for providing all the necessary resources needed for the completion of this work. On a personal note, I would like to express my appreciation to my family and friends for their encouragement and support through this journey.

To my family.

This work was possible because of their strength, wisdom, and love.

Publications

Journal Papers

[J1]. **A. Savva**, T. Theocharides, and C. Nicopoulos, “Robustness of Artificial Neural Networks Based on Weight Alterations Used for Prediction Purposes," *MDPI Algorithms 2023*, 16, 322, 2023.

[J2]. **A. Savva**, T. Theocharides, and C. Nicopoulos, “A Design Space Exploration Framework for ANN-Based Fault Detection in Hardware Systems," *Journal of Electrical and Computer Engineering, Hindawi*, 2017.

[J3]. **A. Savva**, T. Theocharides, and V. Soteriou, “Intelligent On/Off Dynamic Link Management for On-Chip Networks," *Journal of Electrical and Computer Engineering*, vol. 2012, Article ID: 107821, 2012.

Conference Papers

[C1]. **A. Savva**, T. Theocharides, and V. Soteriou, “Intelligent On/Off Dynamic Link Management for On-Chip Networks," *in Proc. IEEE Annual Symposium on ISVLSI*, 2011, pp. 343 - 344.

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Network
CMOS	Complementary Metal Oxide Semiconductor
CNN	Convolutional Neural Networks
CPU	Central Processing Unit
DNN	Dynamic Neural Network
DMR	Dual Modular Redundancy
DRAM	Dynamic Random Access Memory
DVFS	Dynamic Link Voltage and Frequency Scaling
DVS	Dynamic Link Voltage Scaling
GPNoC	General Purpose Simulator for Networks on Chip
IDS	Intrusion Detection System
ITRS	International Technology Roadmap of Semiconductors
LSTM	Long Short-Term Memory
LUT	Lookup Table
MAC	Multiply Accumulate Operation
NN	Neural Network
NoC	Network on Chip
n − MR	n-Modular Redundancy
OCP	Open Core Protocol
ORION	Power-Performance Simulator for Interconnection Networks
PARSEC	Princeton Application Repository Shared Memory Computers
PE	Processing Element
PoPNet	Population Network Simulator
QoS	Quality of Service
RFP	Robustness-Aware Filter Pruning

RISC	Reduced Instruction Set Computer
ROC	Receiver Operating Characteristic
SoC	System on Chip
SRAM	Static Random Access Memory
SYN	Synchronization Packets
USB	Universal Serial Bus
VC	Virtual Channel
VLSI	Very Large-Scale Integration

Chapter 1

Introduction

1.1 General Introduction

Nowadays, with the rapid evolution of hardware, having reliable hardware components has become of great importance. Power consumption, reliability and robustness however, are major limitations in hardware today and researchers have been constantly working on reducing those issues. Monitoring the hardware and providing intelligent solutions has been given significant attention from the research community. For the purposes of this work, Networks on Chip (NoC) will be used as case study.

NoCs provide scalable on-chip communication and are expected to be the dominant interconnection architectures in multicore and manycore systems. NoCs borrowed many concepts and techniques from packet-switched computer networks and they provide high performance and reliability. NoCs scale better than traditional forms of on-chip interconnect and they have better performance characteristics. Based on those characteristics, using NoCs technologies for the future generations has become of great importance. NoCs organize the communication between operating modules located on the chip and ensures maximum data transfer speeds and reduction in the total number of necessary physical connections. NoCs are trend to be an emerging technology which can benefit from the ideas of network systems architecture – switching architectures and will be used as case study for our work.

Some of the most important concerns in designing hardware and networks in general are the power consumption and fault tolerance. For the purposes of proposing intelligent monitoring techniques and providing optimized solutions for these issues a design exploration framework must be developed first and based on this framework new intelligent techniques/mechanisms will be proposed. For the development of the intelligent framework, analysis and evaluation of all the parameters of the NoC with the use of Artificial Neural Networks (ANNs) will be presented and analyzed for high level power management, fault detection and ANN robustness.

While implementing an exploration framework, there are many issues to deal with, which have inspired researcher's attention. Starting with the NoC architecture different parameters should be analyzed and measured for optimal design. It should be noted here that building a simulator for NoCs from scratch it's not a thesis objective and we can rely on an existing simulator which will be altered for the purposes of our work. Additionally, determining the ANN network structure, the number of neurons in hidden/output layers and the procedure of the ANN training are some of those issues. The developed NoC and ANNs are explained and evaluated and are adaptable in many different hardware systems. First step of this work is to identify and analyze all the main steps needed for this framework. The developed exploration framework will help us to develop and evaluate new techniques for the NoCs and ANNs. All the necessary steps for designing such techniques will be analyzed, explained and evaluated.

Power consumption, is a major limitation in hardware and NoCs today and researchers have been constantly working on reducing both dynamic and static power. Motivated by the findings of the previous works for power consumption, we propose the use of Artificial Neural Networks as a dynamic link power consumption management mechanism, by utilizing application traffic information. Based on their ability to dynamically be trained to variable scenarios, ANNs can offer flexibility and high prediction capabilities [1]. An ANN-based mechanism is developed to intelligently compute dynamically which links can be turned off and on during discrete time intervals. The ANN receives link utilization data in discrete time intervals, and predicts the links that should be turned off or on based on appropriate training. ANNs can be dynamically trained to new application information, and have been proven that they can offer accurate prediction results in similar scenarios [2]. ANNs can be efficiently designed in hardware provided they remained relatively small, through efficient resource sharing and pipelining. Furthermore, by partitioning the NoC, individual small ANNs can be assigned to monitor each partition independently, and in parallel monitor the entire network. We introduce topology-based directed-training as a pre-training scheme, using guided simulation, which helps to minimize the large training set and the ANN complexity.

One major problem which affects the reliability in hardware systems is the presence of different types of faults. These faults change the expected behavior of a hardware system and can be temporary or permanent. Temporary faults occur most of the times because of the cross-talks and noise. Permanent faults occur due to manufacturing defects. Fault detection in hardware systems and networks remains one of the top challenges and the development of an intelligent fault-detection scheme is needed. For the purposes of developing an intelligent fault detection mechanism with the use of ANNs through the developed framework, a case-study scenario based on NoCs is used.

In future on-chip generations, there will be a huge increase in faults [3], [4]. According to ITRS (International Technology Roadmap for Semiconductors), in the near future, the manufacturing defect rate will reach approximately up to 1000 defects/m2 [4]. Interconnection faults can potentially create disconnected networks and as a result network will not be able to function properly. Studies show that complex error detection schemes may require high energy dissipation and huge area overheads, which have direct impact on the performance of the systems introducing also extra latencies [5], [6]. The fault detection of the NoCs systems can be done based on the testing of the routers, interconnects and the processing elements. Many efforts have been made in order to detect faults in hardware systems and NoCs which include different fault testing/detection strategies [7]–[9]. Most of them present low level prediction mechanisms. Additionally with the new proposed intelligent fault detection mechanism our work focuses on the analysis of different aspects of the framework for high level fault detection with the use of ANNs. Emphasis is placed on the topological analysis of ANN networks and on the different parameters needed for the design of the ANN for this purpose.

Additionally with the above, neural networks provide state of the art results for most machine learning tasks. Unfortunately, they are vulnerable to changes. This makes it difficult to apply them in critical areas [10]. While neural networks have achieved prediction capabilities, their accuracy drops in the presence of small perturbations. One of the main research goals of the analysis of robustness is to propose different solutions / architectures with increased robustness [11]. This is probably one of the fundamental problems that need special attention from the researchers in the years to come [11]. Till now, researchers were concentrated on comparing results based on having/not having adversarial machine learning. Our work will provide a study for robustness based on weights alterations for ANNs. Discussion of the robustness of neural networks to changes of weights that might affect the prediction results will be presented and a robustness technique for ANNs will be developed.

1.2 Motivation - Goals

Power consumption, reliability and robustness for hardware and networks in general are some of the most important aspects that researchers have to deal with nowadays. In order to be able to propose new intelligent methods/techniques to deal with these issues, an intelligent framework is needed. First part of this thesis will introduce and develop an intelligent exploration framework which will analyze and evaluate all the important parameters of NoCs, which will be used as case study for our work, with the use of Artificial Neural Networks. Our framework will provide an opportunity for the interconnection network evaluations to be done in a full simulation fashion with the optimal parameters. In addition, it provides detailed and accurate NoC and ANN models. Also, it enables the evaluation of new intelligent techniques which use interconnection on-chip networks. Through this framework, new intelligent methods will be presented and evaluated in order to minimize power consumption, increase the error detection accuracy in hardware and NoCs, and increase the robustness of the ANNs. ANNs have shown impressive prediction results based on the appropriate training and they will be used for this work in order to provide intelligent monitoring techniques/methods taking advantage of the characteristics that the ANNs can offer. Based on their training property, ANNs provide a comparative advantage against statical ways, making the framework flexible under any application it is required to facilitate. The main objectives of this thesis are:

O1: Develop an intelligent dynamic power management policy.

- Study how links that connect the NoC routers affect power consumption.
- Develop an intelligent method for minimizing power consumption for hardware and NoCs with the use of ANNs – Intelligent On/Off dynamic link man-

agement for on-chip networks.

- Present the power savings and hardware overheads in comparison with other methods/techniques.

O2: Develop a design – space exploration framework for ANN-based Fault Detection in hardware and NoCs.

- Find optimal parameters for the framework for fault detection purposes.
- Develop decision steps for the fault detection in hardware/NoCs (for the ANNs and the simulator).
- Develop and evaluate an intelligent ANN mechanism for the fault prediction in NoCs/hardware.

O3: Study the robustness of ANNs based on weight alterations and develop a robust technique.

- Study the robustness of ANNs used for prediction purposes based on weights alterations.
- Develop robust technique for ANNs to be included in our work.
- Alter the architecture of the already developed ANNs in order to have high robustness.

1.3 Thesis Contributions

This work provides firstly, a new intelligent technique for the link power consumption management of the hardware/NoCs. Through this technique, NoC links are turned off and on through the use of small and scalable neural networks which intelligently predict the links to be turned off/on. Based on the simulation results with various synthetic traffic models over various network topologies, the proposed method reaches up to 13% more power savings when compared to trivial threshold computation at very low (<4%) hardware overheads. Secondly, a new intelligent method for the detection of faults in hardware is developed, choosing optimal parameters for the framework. Simulation results show that the proposed work detects up to 99% of inter-router link faults with a delay less than 60 cycles. Synthesis results indicate an estimated amount of 0.0633mW power consumption per neuron for the implemented ANN when computing a complete cycle. Lastly, a method for increasing the robustness of the ANNs is developed and evaluated. Based on simulations this technique maintains the robustness of the ANNs at high levels, around 98% prediction accuracy with minimum additional hardware overheads. Additionally with the above, this work provides an exploration framework for ANN based techniques for NoCs to be used as case study. With the use of the framework, the new intelligent methods/techniques mentioned are developed and evaluated. This framework also gives an opportunity to the research community to evaluate new methods and techniques in a full system simulation fashion.

1.4 Thesis Organization

This dissertation is organized in seven chapters, with Chapter 2 presenting the background and state of the art, Chapters 3 up to 6 containing the technical contributions of this thesis (exploration framework, intelligent power management policy for NoCs, high level fault detection technique for hardware and intelligent method for increasing the robustness of ANNs with minimum additional hardware overheads), and Chapter 7 summarizing the thesis' findings and laying out possible future research directions. The codes for the NoC and ANN can be found in [12] and [13] respectively. A detailed description of each chapter is presented below.

Chapter 2 provides background information regarding NoCs and ANNs technologies and the different methods used for dynamic power management, fault detection mechanisms for hardware and NoCs, and lastly robustness methods and techniques for ANNs. Specifically, technologies and techniques used for NoCs and hardware are discussed, including Dynamic Voltage Frequency Scaling (DVFS) and static threshold-based techniques for the link power management. For the fault detection different techniques with diagnostic abilities are presented and lastly, robust techniques like n-MR (Modular Redundancy) schemes and techniques based on adversarial examples are presented for the robustness of ANNs.

In Chapter 3, the experimental framework is presented and discussed, where by using the proposed framework, new methods and techniques will be developed and evaluated. Through this chapter, the design of the exploration framework is presented along with the different processing units which are added in the framework. Further, in this chapter, the ANN development is discussed with the different ANN parameters and the different decisions for the ANNs and the simulator.

In Chapter 4, the use of the ANNs for intelligent dynamic link management is proposed. Through this approach, the advantages of the ANNs are used in order to intelligently predict which links should be turned off and on based on the system status (link utilization levels). Links are turned off and on via the use of scalable ANNs. The specific mechanism is presented and compared against non-intelligent case and other related literature works showing better power savings while having lower hardware overheads. This technique developed in chapter 4 with all the different simulation results and conclusions were published in [C1 and J3].

Further, the fault detection in hardware is considered in Chapter 5. In this chapter, an intelligent approach based on the use of ANNs is proposed which intelligently detects future faults. Additionally, this chapter provides analysis for the process of creating an exploration framework for fault detection purposes with optimal parameters in hardware systems. The framework and fault detection technique developed in Chapter 5 were published in [J2].

Chapter 6 presents a study based on simulations for the robustness of ANNs used for prediction purposes based on weights alterations. Additionally, through this chapter, a robust method for ANNs is developed and evaluated based on redundancy. Through this method, only the most important neurons/connections are replicated. The work presented in Chapter 6 was published in [J1].

Lastly, Chapter 7 summarizes the findings of this dissertation and discusses interesting future research directions.

Chapter 2

Background

In this chapter, background information regarding NoCs, ANNs and different techniques for power consumption, fault tolerance and ANN robustness are discussed. NoCs and ANNs are very popular and active research topics in the literature. Various studies are done concerning both of them. Through this literature review firstly we are going to present works about ANNs and their uses for prediction purposes. ANNs have shown impressive prediction results based on the appropriate training. For the purposes of this thesis, these prediction capabilities will be used in order to develop new intelligent techniques for the NoCs and hardware. Based on our work, the size of the developed ANNs will be kept to minimal and the complexity will be minimized as well (- will be explained in the next chapters). This will help to keep the overheads and additional hardware resources to minimum and make optimal use of ANN's capabilities. Table 2.1 presents a summary of the different related works categories covered in this chapter.

Literature review which follows will also cover works for power consumption management and fault detection in NoCs. Although NoCs have many advantages there is still the issue of high-power consumption due to the routing requirements. For the purposes of our work, ANN prediction capabilities will be used to intelligently minimize the power consumption of the links based on the use of the ANNs characteristics mentioned above. Additionally, ANNs will be used for fault detection in NoCs and hardware providing new intelligent method based on their prediction capabilities. At the end of this section, literature about robust techniques for ANNs will be presented and analyzed. While ANNs have achieved prediction capabilities, their accuracy drops in the presence of small changes and this might affect seriously

Table 2.1: Description of the different work categories.

Related Work Category	**Description / Summary**
1. ANNs and Prediction Techniques	Related works about ANN model approaches for prediction purposes are presented, showing the importance and the predicting abilities of the ANNs.
2. Link Dynamic Power Management Techniques	Related works for Link Power Consumption are presented like: Dynamic Voltage and Frequency Scaling techniques, software-based techniques for power management and new run-time techniques.
3. Fault Detection Techniques	Related works for Fault Detection are presented like: new frameworks for fault diagnosis based on new routing algorithms and works which introduce methods for fault prediction based on the use on Neural Networks.
4. Robust Techniques	Related works for Robustness are presented like: n-MR schemes with redundancy, techniques based on certifications for the robustness and techniques based on adversarial attacks.

critical decisions. Robustness of ANNs is one of the most important issues that researchers need to address in the years to come. Our work will provide intelligent method to increase the robustness of the ANNs.

2.1 ANN and Prediction Related Work

The ANNs are considered to have excellent characteristics such as generalization capability, robustness and fault tolerance [1]. The neural networks are able to handle large input data sets and based on appropriate learning, they are able to find complex non-linear relationships among the data in order to make accurate predictions. Significant efforts have been made in order to develop a prediction framework for different cases based on ANNs [14]–[17]. Based on these, ANNs offer high prediction capabilities. ANNs also have been successfully applied in various real-life scenarios which include learning systems [18], neuroscience [19] and engineering [20]. Through these, it is believed that ANNs are a powerful tool which has the ability to make predictions based on complex relations of the input and output data. Motivated by these findings, this work proposes a framework which uses ANNs for power management, fault detections and robustness. Integrated hardware-based ANNs are used, and based on the appropriate ANN training they intelligently make the detection.

There are many approaches to develop ANN models for real life problems which state the importance of ANNs. ANNs have been used as branch prediction mechanisms in computer architecture, as forecasting mechanisms in price prediction of Share-Market [17]. According to Shamishi et al. [21], it is explained how Matlab tools can be used in writing scripts, which will help the development of ANN models in order to predict global solar radiation in United Arab Emirates. Added to this, Jahirul and Suzuki et al. [14], [16], provide advances of ANN applications on different situations. They present the methodology and the biomedical applications of ANNs as well as applications of ANNs in industry and engineering.

A lot of work has been done in the research area for fault prediction in high level systems with the use of ANNs. An ANN is a mathematical model which simulates the structure and functionalities of biological neural networks [1]. The functionality of an ANN can be represented in three basic steps. At the first step, the inputs of the ANN are weighted. This means that the inputs are multiplied with appropriate weights. During the second step, the summation of all the weighted inputs is calculated. At the end, this summation of the previously weighted inputs passes through the activation function. The neuron output is then propagated to the neurons of the next layer which perform the same operation with the newly set of inputs and their own weights. This is repeated for all the layers of an ANN [22].

Next sub-section presents some of the previous works done for power management in Networks on Chip since in the future ANNs can be used for this purpose.

2.2 Link Dynamic Power Management

Research practice surveys such as [23] which outline the design challenges and lay the roadmap in future NoC design have emphasized the critical need to conduct research in NoC power management due to concerns of battery life, cooling, environmental issues, and thermal management, as a means to safeguard the scalability of general-purpose multicore systems that employ NoCs as their communication backbone. Link dynamic power management has been given significant attention by NoC researchers, as circuit-based techniques such as differential signals and low-voltage swing hardware using level converters do not seem to adequately address the power management problem [24], [25]. As such, there is a significant shift towards high-level techniques such as selective turning of links on and off. The challenge involved in those techniques, includes the computation of the decision on whether a certain link is to be turned off, and when it will be turned back on. These decisions typically rely on information from the system concerning link utilization, and, so far, have been taken using a threshold-based approach. There have been attempts in dynamic link frequency and dynamic link voltage (DVFS) management with most using these thresholds as well.

Among the proposed techniques, some approaches use software-based management techniques such as the one in [24], which proposes the use of reducing energy consumption through compiler-directed channel voltage scaling. This technique uses proactive power management, where application code is analyzed during static compilation-time to identify periods of network inactivity; power-management calls are then inserted into the compiled application code to direct on/off link transitions to save link power. A similar approach was also taken in [26] for communication power management using dynamic voltage-scalable links. Both of these techniques, however, have been applied to highly predictive array-intensive applications, where precise idle and active periods can be extracted. Hence run-time variability, applicable to NoCs found in general-purposed multi-core chips, has not been examined. Further the work in [27] proposes software-hardware hybrid techniques that extend the flow of a parallelizing compiler in order to direct run-time network power reduction. In this work the parallelizing compiler orchestrates dynamic-voltage scaling of communication links, while the hardware part handles unpredicted online traffic variability in the underlying NoC to handle unexpected swings in link utilization that could not be captured by the compiler for improved power savings and performance attainability.

Low-level, hardware-based techniques that determine on/off periods and manage the voltage and frequency, exhibit however better energy savings as they can shorten the processing time required for a decision whether to turn a link off or on to be made. The most commonly used power management policies deal with adjusting processing frequency and voltage (Dynamic Voltage Scaling - DVS). The works in [25] and [26] present DVS techniques that feature a utilization threshold, to adjust the voltage to the minimum value while maintaining the worst-case execution time. In [28] the authors propose that the dynamic voltage scaling is performed based on the information concerning execution time variation within multimedia streams. The work in [29] proposes a power consumption scheme, in which variable-frequency links can track and adjust their voltage level to the minimum supply voltage as the link frequency is changed. Furthermore, [30] introduces a history-based DVS policy which adjusts the operating voltage and clock frequency of a link according to the utilization of the link/input buffer. Link and buffer utilization information are also used in [31], which proposes a DVS policy scheme that dynamically adapts its voltage scaling to achieve power savings with minimal impact on performance. Given the task graph of a periodic real-time application, the proposed algorithm in [31] assigns an appropriate communication speed to each link, which minimizes the energy consumption of the NoC while guaranteeing the timing constraints of real applications. Moreover, this algorithm turns off links statically when no communications are scheduled because the leakage power of an interconnection network is significant. In general on/off links have, in most cases, been more efficient than DVFS techniques, as links, even if operating at a lower voltage, still consume leakage and dynamic power [32], [33]. These works therefore present a threshold-based technique that turns links off when there is low utilization, using a statically computed threshold. Given that static computation by nature is pessimistic, dynamic policies have been proposed. Research work in [34] proposes a mechanism to reduce interconnect power consumption that combines dynamic on/off network link switching as a function of traffic while maintaining network connectivity, and dynamically reducing the available network bandwidth when traffic becomes low. This technique is also based on a threshold-based on/off decision policy. Next, the work in [35] considers a 3-D torus network in a cluster design (off-chip interconnection network) to explore opportunities for link shutdown during collective communication operations. The scheme in [36] introduces the Skip-link architecture that dynamically reconfigures NoC topologies, in order aiming to reduce the overall switching activity and hence associated energy consumption. The technique allows the creation of long-range Skiplinks at runtime to reduce the logical distance between frequently communicating nodes. However, this is based on application communication behavior in order to extract such opportunities to save energy. Finally, the related work in [37] explores how the power consumed by such on-chip networks may be reduced through the application of clock and signal gating optimizations, shutting power to routers when they are inactive. This is applied at two levels: (1) at a granular level applied to individual router components and (2) globally at the entire router.

Run-time link power management has recently gained ground in research to address the leakage issues as well. As links become heavily pipelined to satisfy performance constraints, link buffers and pipeline buffers contribute significantly in leakage power consumption. As such, the problem becomes significant with the increased on-chip NoC sizes, impacting both the power consumption, as well as the thermal stability of the chip. Dynamic link management techniques have therefore been proposed; the work in [38] proposes an adaptive low-power transmission scheme, where the energy required for reliable communications is minimized while satisfying a QoS (Quality of Service) constraint by varying dynamically the voltage on the links. The work in [39] introduces ideas of dynamic routing in the context of NoCs and focuses on how to deal with links or/and routers that become unavailable either temporarily or permanently. Such techniques are a little more complicated than a threshold-based approach, and inhere performance overheads during each dynamic computation. As such, the work in [40] introduces the idea of an intelligent method for dynamic (run-time) power management policy, utilizing control theory. A preliminary idea of a closed-loop power management system for NoCs is presented, where the estimator tracks changes in the NoC and estimates changes in service times, in arrival traffic patterns and other NoC parameters. The estimator then feeds any changes into the system model, and the controller sets the voltage and frequency of the processor for the newly estimated frequency rate.

Motivated by the promising results presented in [40], and the potential performance benefits of dynamic threshold computation techniques, our work proposes a new dynamic, intelligent and flexible scheme based on ANNs for dynamic computation of the threshold that determines which links can be turned off or on.

Next sub-section presents some of the previous works done for fault detection in Networks on Chip.

2.3 Fault Detection in Networks on Chip

Sanaye et al. [41], present a new approach based on ANN's implementation for fault detection/phase selection for transmission lines. Neural networks in this work are used in a protective pattern classifier algorithm. The proposed algorithm implement fault detection, classification and fault phase selection for transmission lines. Authors in [42] have presented a new method for detection high impedance faults in electrical distributed systems with the use of ANNs. The proposed neural network was trained and tested based on simulation data from different system conditions and implemented on a digital signal processor board.

Moreover, authors in [43] propose NoCAlert, an online fault detection mechanism which is based on the idea of invariance checking. Through the invariance checking, the outputs of the control logic modules of the NoC are checked for wrongly outputs based on the current inputs (micro checker models in hardware). In [44], authors propose uDIREC, a unified framework for permanent fault diagnosis based on the use of a deadlock free routing algorithm which helps the working links in the NoC to be maximally utilized in case of fault. uDIREC finds reliable routes which use the links that are still working in the NoC. Added to this, in [45], authors propose Hermes, a fault tolerant routing algorithm for NoCs which is deadlock free. Hermes balances the traffic in order to achieve higher performance for fault free paths and at the same time provides pre-configured paths in case of faults.

The work in [22], presents an intelligent power management policy for networks on chip where links are turned off and switched back on, based on ANNs predictions. The ANNs use the link utilizations as feedback from the system and based on these, they select candidate links for turning off in an effort to achieve power savings in NoCs. All the aforementioned works present the ANN models for prediction purposes.

Our work, presented in the next chapters, uses ANNs for detection purposes and presents how different parameters are used in designing an ANN-based fault detection framework. Added to this, it takes into consideration all the necessary constraints like extra hardware overheads, accuracy, speed, latency. Motivated from previous works and the ideas in [22], our work creates as a case study scenario an intelligent framework for inter-router link fault detections in NoCs and explains how integrated small sized hardware based ANNs can be used for this purpose.

Next sub-section presents some of the previous works done for robustness of ANNs.

2.4 Robust Techniques for ANNs

ANNs and robustness are very popular and active research topics in the literature. Due to adversarial attacks, many approaches have been developed regarding the fault tolerance and robustness of ANNs. The authors of [46] proposed a method based on neural networks in order to detect malicious hosts based on the SYN packets that are exchanged. With the aid of appropriate training, this method achieved 98% accuracy based on specific test data. Moreover, the authors of [10] found that powerful attacks can defeat defensive distillation, demonstrating that by systematically evaluating several possible attacks, better adversarial examples can be found than those in existing approaches. This study also concludes that constructing defenses that are robust to adversarial examples remains challenging. The study of [11] presents a survey on the robustness of deep networks to changes that may affect the samples in practice, such as adversarial perturbations, random noise, and transformations. The authors also discuss different solutions that attempt to increase the robustness of deep networks. Additionally, the authors of [47] study the effectiveness of different types of attacks and propose methods for training a deep-learning-based ID (Intrusion Detection System) with the use of different types of neural networks in order to increase the robustness of the networks based on a thread model.

Furthermore, n-MR (Modular Redundancy) schemes and redundancy have been proposed as methods for increasing the robustness of ANNs. The authors of [48] propose a novel dual modular redundancy framework for DNNs (Deep Neural Networks). D2NN checks the fault sensitivity of each neuron in the target DNN based on performance degradation and shows if the neuron is faulty. Next, D2NN duplicates the more sensitive neurons to construct the completed DMR (dual modular redundancy).

Studies that provide certifications for the robustness of neural networks are presented next. The authors of [49] present a defense method for neural networks with one hidden layer. This is based on certifications that, for a given network and test input, there is no attack that can force the error to exceed a certain threshold: the computation of an upper bound in the worst-case loss scenario. Additionally, they optimized this method with different network parameters, providing an adaptive regularizer that helps robustness. The authors of [50] studied the sensitivity of neural networks to weight perturbations. They proposed an efficient approach to compute a certified robustness bound of weight perturbations within neural networks that do not have erroneous outputs. The authors provided a certified weight perturbation region such that DNNs maintained their accuracy if weight perturbations were within that region.

Lastly, different novel robust techniques for neural networks have been presented. The authors of [51] introduced E2CNNs, a new design methodology to improve robustness against memory errors in embedded systems. This work proposes a heuristic method to automate the design of a voter-based ensemble architecture. This design methodology increases the error robustness of CNNs (Convolutional Neural Networks) by using ensemble architectures. The authors of [52] studied the sensitivity of weight perturbation in neural networks and its impact on model performance. They further designed a new theory-driven loss function for training generalization and robust neural networks against weight perturbations: bounded weight perturbations. Moreover, the authors of [53] extended the definition of robustness to any type of input for which some alterations can be defined. They proposed the ROBY tool, which accepts different types of data, and some alterations can be performed on these data providing the ability to classify the input data correctly. The authors of [54] present a new scheme for robust DNNs called coded

DNN. This alters the internal structure of DNNs by adding redundant neurons and edges to increase reliability—a new middle layer is added. The authors of [55] proposed an approach that is complementary to other forms of defense and replaces the weights of individual neurons with robust analogs derived from the use of Fourier analytic tools. Additionally, the authors of [56] propose a new method called robustness-aware filter pruning (RFP) and utilize this filter pruning method to increase the robustness against adversarial attacks. In the proposed method, the filters that are involved in non-robust features are pruned. Lastly, the authors of [57] designed a novel neuron that uses L distance as its basic operation, known as an L-dist neuron. They show that the L-dist neuron has a natural 1-Lipschitz function with respect to the L norm, and the neural networks constructed with this neuron (L-dist Nets) have the same property [57]. This directly provides a theoretical guarantee of the certified robustness based on the margin of the prediction outputs.

Motivated by the above related works for robustness, our work presents a thorough study based on simulations for the robustness of neural networks based on weights alterations. Based on this study, a new robust technique is presented for ANNs with minimum additional hardware overheads.

Chapter 3

Experimental Framework

The first step is to establish a framework for developing and evaluating intelligent methods/techniques with the use of ANNs. This framework will help us and the research community to develop and evaluate new techniques in a full intelligent simulation fashion. Additionally, detailed NoC and ANN models will be developed through the framework and will be adaptable in different hardware systems and networks. Through this framework, new intelligent methods/techniques for power management, fault detection and robustness will be implemented and evaluated. The two most important parts of the framework are of course the NoC and ANNs. In order to develop this framework, we started from the creation of the NoC simulator and then we moved with the implementation of the different components of the new system. Added to this, we developed the appropriate ANNs in order to have a complete intelligent simulation and evaluation framework for many-core architectures which will be used for the evaluation of the new proposed techniques.

3.1 Designing Exploration Framework for NoCs

A complete NoC architecture consists of three main parts: 1) input output data models, 2) the connecting interface where in NoCs it consists of the routers which are connected with each other in order to transfer the data, and 3) the processing units (PEs) like processors, control units, memory which are following the Open Core protocol (OCP) in order to add them easily in the overall system. Figure 3.1 shows the main parts of a NoC Architecture. PEs are the Processing Units; R stands for the routers and the connection lines between the routers represents the links.

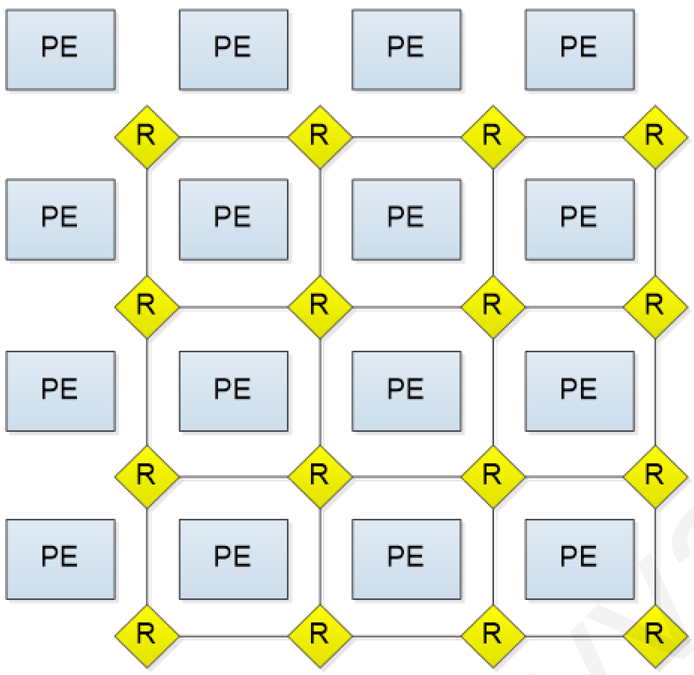


Figure 3.1: NoC Architecture.

At the beginning we started the implementation by studying and analyzing different already implemented simulators which satisfy the requirements of a many-core system. Two of them were very close to the requirements of many-core systems. Those are: Pop-Net (Population Network Simulator) in C++ programming language and gpNoCsim (general purpose NoC simulator) simulator implemented in Java programming language.

We chose to work with gpNoCsim simulator because it has the high level principles of implementation of a framework and also is more flexible in programming compared with Pop-Net simulator. Added to this, we implemented core models, memory models as well as input/output units for the data. We managed to implement a simulation and evaluation framework, in which new units can be added in order to be evaluated based on power consumption, throughput, latency and reliability of the system. After the completion of the changes/additions in the simulator, we managed to implement virtual channels as well as priority conditions for our data.

Firstly, we present a general purpose modeling and simulation framework for NoC in gpNoCsim — an open-source, component based network simulation environment that is developed entirely in Java. This framework is built upon the object-oriented modular design of the NoC architecture components and will be

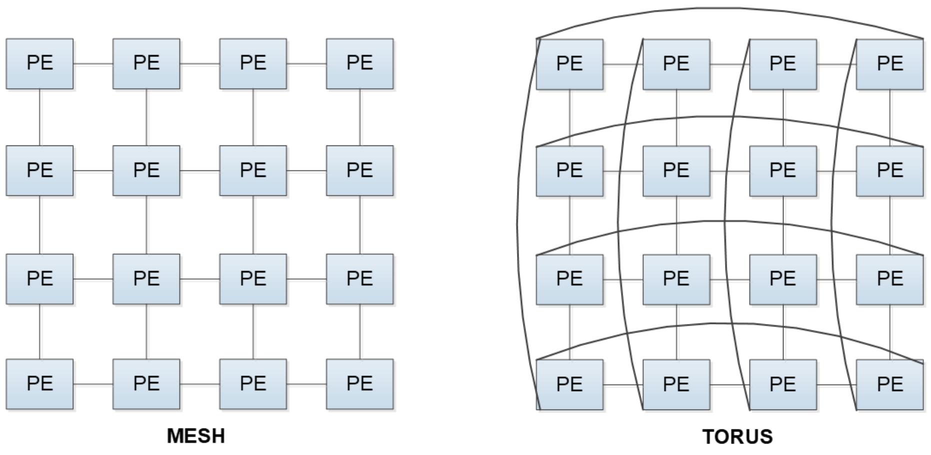


Figure 3.2: Mesh and Torus Architectures.

used to evaluate our new techniques. We demonstrate the use of proposed framework, i.e. gpNoCsim, by simulating several existing, well-known architectures. In addition, we have provided the guidelines to simulate future architectures by making simple modifications. As a general purpose simulator for the NoC architectures, gpNoCsim provides the designers with the flexibility of incorporating different network and traffic configurations. Network configuration has been considered as the conglomeration of topologies, switching techniques, and routing algorithms. Reconfigurable topologies allow comparison between different architectures: Mesh, Torus, Butterfly, Fat Tree. Figure 3.2 shows the Mesh and Torus architectures.

3.2 Processing Units

With the development of the simulation framework, we created the processing unit models which will be added in the network framework in order to complete the network simulator. The processing units are just the units which are added as terminals into the network and they are responsible for processing the data. For this work, three main categories of processing units are used: processor cores, memory units and input/output controllers and control units.

The cores which are implemented are simple based on the RISC (Reduced Instruction Set Computer) philosophy and they are fast. 32-bit and 64-bit models are created in order for the user to choose the system he wants to evaluate. For the addition of cores into the simulator framework the Open Core Protocol was followed, which gives the user the potential of easy addition in the system already made cores which are used in the industry. Next, the memory units are created which are autonomous (they have the memory controllers) and they follow the Open Source Protocol. SRAM (Static Random Access Memory) memory models are developed which is the current technology for memory for NoCs. Added to this, the memory models include error control units and coherency units for the purposes of correct data when are used in parallel from two or more cores. Based on the above, the user can create one centralized memory which can provide data to the cores or he can create many independent distributed memory units into the system.

Input/output data models are also developed which are communicating with the network based on the communication protocol of the network but the user can give the parameters for communication with the outside world in order for the simulation framework to communicate with different units outside the NoC like external DRAM (Dynamic Random Access Memory), Ethernet gates and USB (Universal Serial Bus).

Based on the above, we managed to model different units which are crucial for many-core systems and they will be used for the development of a general system for simulating many-core systems.

3.3 General for NoC Simulator

Generally, NoCs are the basis for the communication between the cores, the memory units and the rest of the system units. One network consists from the routers, the links which connect the routers between each other as well as the routers with the processing units (PE) and the input output units for the processing data. Added to these, one network can have control units for the correct transfer of the data. Many parameters are needed for the simulation and evaluation of a network and they have to deal with the network topology (mesh, torus, ring), the routing algorithm (X-Y, adaptive, hierarchical), the size of data packets sending through the routers and many more.

In order to create the NoC architecture, we developed a simulation framework based on the Java-based cycle-accurate gpNoCsim simulator (General Purpose Simulator for Network – on – Chip Architectures) [58]. The framework enables simulation of multiple topologies, utilizing adaptive dimension-ordered XY routing algorithm with virtual channel support and 4-stage pipelined router operation. We experimented with the adaptive XY routing algorithm which induced blocking. The simulated router supports 64-bits flit width, 4 virtual channels per link and two buffers per virtual channel. The routers used are all the same, and we assume wormhole flow control. The framework supports various synthetic, realistic and user-defined traffic models. We assumed an 8x8 mesh topology consisting of 64 routers. Simulations will run for this topology in order to collect training data for the ANNs. Every x cycles, new training data will be sent to the ANNs for training. For scalability purposes, partition of the NoC into smaller regions is needed and an ANN is assigned to be responsible for each region. This partition will help to keep the ANN sizes relatively small, reusable and easy to implement in hardware. The ANN mechanism can be considered as a different independent PE, on top of the NoC topology. Different NoC partitions can be implemented and analyzed in order to choose which one is appropriate for the monitoring framework. NoC topology and ANNs are presented and analyzed in details in the next chapters.

Simulations are done over a range of 1,000,000 cycles, with a warm-up period of 100,000 cycles. In the 8×8 topologies, we partitioned the NoC into regions, where each ANN-based model was assigned as responsible for monitoring. The ANN-based models monitored all the routers and links in their corresponding partition and all ANN results related to the size and operation of the ANN are given based on these architectural details.

Time was divided into x-cycle intervals; at the end of each interval, all routers in the NoC partition transmit their average utilization data for that span (computed via a counter and LUT (Lookup Table)-based multiplication with the reciprocal of the interval). A time-out mechanism equal to the expected delay of each router towards the ANN mechanism is imposed, to maintain reasonable delays. The ANN receives one packet from each router with four utilization values, one for each port. The ANN then proceeds with the prediction mechanisms.

For the power modeling, we adopted the Orion power models for the dynamic power consumption of each router [59]. Router and link hardware were designed and synthesized in Verilog and Synopsys Design Compiler in order to obtain the leakage power values. We used a commercial CMOS (Complementary Metal-Oxide Semiconductor) 65nm library, and a sequence of random input vectors, for several thousand cycles, and measured the leakage power of each router and link, through all computation cycles and combinations of events. The leakage values are then fed into the simulator, along with the Orion models for active power, and the overall power is computed.

After the development of the simulator, four different traffic models were implemented and checked and they are going to be used for evaluation purposes of the network along with realistic traffic profiles. Uniform-Random Traffic: In random traffic, each source is equally likely to send to each destination - is the most commonly used traffic pattern in network evaluation. Random traffic is very benign because, by making the traffic uniformly distributed, it balances load even for topologies and routing algorithms that normally have very poor load balance. Transparent Traffic: Traffic here is created with the use of matrix transpose or corner – turn operations. Tornado Traffic: Based on digit permutations in which the digits of the destination address are calculated from the digits of the source address. The tornado pattern is designed as an adversary for topologies, whereas modeling traffic measures a topology's ability to exploit locality. Neighbor Traffic: Traffic here is created based on the neighboring nodes. Additionally, PARSEC (Princeton Application Repository Shared Memory Computers) benchmark suite which is composed of multi-threaded emerging workloads is used for realistic traffic profiles. These traffic models are added in the simulation framework. Table 3.1 shows the key characteristics of the PARSEC benchmarks used for this work [60]. The routing model has the routing unit, virtual channels, buffers, crossbar switch and the arbitration unit. The developed simulator is scalable and based on that, the user can add new topologies and routing algorithms easily.

For the implementation of the framework, Artificial Neural Networks are used for the proposal of intelligent methods/techniques and for evaluation purposes (next chapters). Based on these we managed to evaluate the implemented framework with regards to performance, reliability and the power consumption. Next, the ANNs development is presented.

3.4 General for ANNs Development

ANNs are able to find complex and non-linear relationships among the data in order to make accurate predictions. For this work, integrated hardware based ANNs are used and based on the appropriate ANN training and the received utilization values in discreet interval times intelligently help the fault detection, power management and robustness processes. ANNs are developed in MATLAB with the use of nntool.

Table 3.1: Summary of the key characteristics of PARSEC benchmarks used for the purposes of this work.

Program	**Application Domain**	**Working Set**	**Data Usage: Sharing / Exchange**
blackscholes	Financial Analysis	small	low/low
bodytrack	Computer Vision	medium	high/medium
canneal	Engineering	unbounded	high/high
facesim	Animation	large	low/medium
ferret	Similarity Search	unbounded	high/high
streamcluster	Data Mining	medium	low/medium
swaptions	Financial Analysis	medium	low/low
vips	Media Processing	medium	low/medium
swaptions	Financial Analysis	medium	low/low
x264	Media Processing	medium	high/high

For the purposes of this work, different kinds of ANNs are created, one for each individual traffic pattern used for the prediction purposes – Random, Tornado, Transpose, Neighbor, PARSEC. The ANNs have the same structure/architecture, they differ only at the input - hidden layer and hidden – output layer weights. Each ANN for example has 20 input neurons in the input layer, one hidden layer with 19 neurons and 20 neurons in the output layer. Each different kind of the four ANNs has its own weights which represent the appropriate traffic pattern.

Each feed-forward ANN follows a fully connected perceptron model. The activation function used is hyperbolic tangent which is symmetric and asymptotic. Hyperbolic tangent produces outputs in scale of [-1, 1] – it is continuous function. Activation function process the output of each neuron for feeding it to the next adjacent layer or network. A network input function gives the summation of all the weights and inputs feeded to the network and afterwards the activation function maps a schema to be feeded to the adjacent layer following it.

The training stage can be done off-line based on the back-propagation ANN training algorithm. Through the first step, inputs of the ANN are weighted (multiplied with the appropriate weights – different for each traffic pattern). Second step, summation of all the weighted inputs of the neuron is calculated. At the end, this summation passes through the activation function. The neuron output is then propagated to the neurons of the next layer which perform the same operation with the newly set of inputs and their own weights. This is repeated for all the layers of the ANN.

The neuron operation can be designed efficiently in hardware since it can be modeled as multiply-accumulate operation (MAC). Each neuron is implemented as MAC unit with the accumulators being multiplexed for each neuron so the number of multipliers is minimized.

The ANN mechanism can be considered as an independent processing element in the NoC (PE). Each base ANN mechanism is responsible for a specific network partition. The ANN mechanism monitors all the link utilization values in the region that is responsible for, and these values are then processed by the ANN.

The size of the ANN is kept relatively small and the ANN complexity is minimized as well as the monitoring of the entire hardware system is done in parallel and independently by each ANN. Although experiments are performed on small networks, the results are considered indicative for larger networks due to the scalable nature of calculations.

3.5 Framework Methodology

Many different decision steps must be taken in order to choose the optimal parameters for developing correctly the framework. Table 3.2 shows the decision steps needed for the general framework from the ANN perspective, while Table 3.3 presents the decision steps needed for the simulator. ANNs can be used in any hardware system. Starting, a decision step is needed for the partition of the network into smaller regions. Individual small-sized ANNs can be assigned to monitor each partition. Following that, decision steps for the architecture of the ANNs must be taken (neurons in input-hidden-output layers, the training of these ANNs) as well as decision steps for different simulation parameters like the sampling period, which is another very important parameter since it is needed for the delay model of the simulator.

In order to design an efficient framework based on ANNs for detection purposes, the ANN architecture as well as the simulation framework have to be analyzed.

Table 3.2: Decision steps for the exploration framework for the ANNs.

ANN Decision Step	**Note**
1. Topology Exploration	Develop in the framework a base network topology – Collect training data for the ANN.
2. ANN Scalability	Experiment with different partitions to choose which one is better.
3. Parameters of the ANN - Training	Find based on experiments efficient parameters for the development and training of the ANN.

Table 3.3: Decision steps for the framework.

Framework Decision Step	**Note**
1. Simulation / Evaluation	Experiment with different sampling periods to find which one gives better results.
2. Delay Model	Find all the extra delay parameters which will be added in the simulator.

This work is focused firstly on the simulator for the collection of training data for the ANNs and also for the simulation results (detection vs. delay) at the end. In addition, the ANN topology and architecture are analyzed, paying more attention on the design of the ANN, the training phase and the detection delay.

Many experiments are made for the completion of the steps of this work. Those are presented in the next chapters. At first, starting from the nxm network, this work chooses a good partition, based on experiments. The architecture of the ANN is studied and a developed ANN is assigned for each created partition. Based on more experiments and appropriate ANN training, decisions about the number of hidden neurons as well as output neurons are taken. Based on these decisions, the ANNs are implemented for the purposes of detecting inter-router link faults, power management and robustness. Next, experiments are needed for the simulator in order to optimally decide different important parameters such as the sampling period. Moreover, a new delay model is added to the simulator which shows the total delay of the detection for comparison purposes.

3.5.1 ANN Training

The ANN mechanism operates in two steps, training step and detection step. In order to train the ANNs correctly, the utilization values collected from the NoC simulation are used. The ANN receives the link utilization values of all the router ports of the partition that monitors. Each router keeps a counter which is used for tracking the travelling packets on each link. If a router fails to transmit its values, then the counter value is set to a sentinel value, indicating that the buffers of that router are fully utilized / blocked. The ANN then uses these utilization values for training. A neural network can be implemented in hardware by using multiplier-accumulator (MAC) units, and a look-up table (LUT) as activation function [22].

The ANN training stage can be performed off-line when the NoC is not used and the training weights can be stored in SRAM based LUTs for fast and on-line reconfiguration of the network. The network is trained with the use of application traffic patterns, off-line, and any ANN training algorithm can be used. In our experiments we used synthetic traffic patterns and realistic traffic profiles (PARSEC benchmarks) and the Matlab ANN toolbox; the weight values were fed to the simulator as inputs, where the actual prediction was then implemented and simulated.

Additional decision steps with simulations and results for the simulator and the ANNs are presented throughout the next chapters.

3.5.2 Simulation Decisions

Another important decision that needs to be examined is the sampling period. Sampling time is divided into different cycle intervals and the experimental results were studied. At the end of each interval, all routers in the partition transmit their average utilization data. The ANN then receives the utilization values and proceeds to the detection.

In order to find a good sentinel value, which will be used when a router fails to transmit its values to the ANN, different simulations for different sentinel values are created.

All these decision steps are presented and analyzed based on simulation results. Based on the above implemented framework different intelligent techniques will be proposed and evaluated for power management, fault detection and robustness. These are presented and analyzed in the next chapters.

Chapter 4

Intelligent Dynamic Link Management for Hardware/NoCs

4.1 Introduction

Links connecting the NoC routers are among the components that consume lot of power. Several attempts have been made to reduce the link power consumption at both the circuit level and the system level. This chapter presents an intelligent dynamic power management policy for NoCs with improved predictive abilities based on supervised online learning of the system status (i.e. expected future utilization link levels), where links are turned off and on via the use of a small and scalable neural network. Simulation results with various synthetic and realistic traffic models over various network topologies are implemented for evaluation purposes.

4.2 Power Management for on-chip Interconnects

Power management is a crucial element in modern-day on-chip interconnects. Significant efforts have been made in order to address power consumption in networks-on-chips [32], [33], [38]. One of the most power-hungry NoC components are the links connecting the routers to each other and the processing elements of the on-chip interconnection network. Data from Intel's Teraflop NoC prototype [61], suggests that link power consumption could be as high as 17% of the network power, and could be even more given the types of links used as well as the size and pipelining involved in designing the link structure. These links, which can be designed with differential signals and low-voltage swing hardware using level converters as circuit-based optimizations for low power consumption, are almost active all the time, even when not transmitting useful data thus spending energy when no inter router communication exists. While such traditional hardware design techniques have contributed towards reducing the power of these links, a system-level technique becomes necessary for more efficient power reduction, as the number of links increases with the scaling and increasing sizes of NoCs, and as application-specific knowledge becomes available. For example, power-aware encoding techniques [62] such as Gray coding cannot be efficiently used, as the hardware cost in the encoder/decoder increases drastically as the system scales to a higher number of network components.

As such, recent research focuses on turning links off and on in order to reduce power consumption, and has been adopted by several works [30]–[33], [40], as certain links in the system are severely underutilized during a specific operational time frame [32]. Techniques such as DVFS (Dynamic Voltage and Frequency Scaling) applied to the link hardware, [30], [31] have been used to vary the link frequency and power according to link utilization, however, even when not data is sent across a link, static power is still being consumed, especially in multi-pipelined links with pipeline buffers in place. In addition, CMOS technology scaling is pointing towards an increased portion of the allocated power budget being consumed as static energy instead of dynamic energy; hence switching on/off links instead of just selectively reducing their frequency/voltage levels offers better power saving advantages as links still do burn power even at lower (i.e. non-zero) voltage-frequency settings [22]. The majority of these on/off link dynamic power-management works employ traditionally a statically-computed threshold value on the link utilization, and based on that threshold value, the link is turned off for an amount of time and then is turned back on when the algorithm decides so. This of course is a pessimistic approach by nature, and imposes harder performance constraints. Recently, the use of control theory for managing candidate links for turning off has been proposed as an idea in [40], with promising results when compared to the statically-based approaches.

Motivated by the findings in [40], this work proposes the use of Artificial Neural Networks as a dynamic link power consumption management mechanism, by utilizing application traffic information. Based on their ability to dynamically be trained to variable scenarios, ANNs can offer flexibility and high prediction capabilities [1].

An ANN-based mechanism can be used to intelligently compute dynamically which links can be turned off and on during discrete time intervals. The ANN receives link utilization data in discrete time intervals, and predicts the links that should be turned off or on. ANNs can be dynamically trained to new application information, and have been proven that they can offer accurate prediction results in similar scenarios [2]. ANNs can be efficiently designed in hardware provided they remained relatively small, through efficient resource sharing and pipelining. Furthermore, by partitioning the NoC, individual small ANNs can be assigned to monitor each partition independently, and in parallel monitor the entire network. This work also introduces topology-based directed-training as a pre-training scheme, using guided simulation, which helps to minimize the large training set and the ANN complexity.

4.3 ANN-Based Threshold Computation Methodology

4.3.1 Static Threshold Computation for Off/On Links

The first step in realizing the proposed ANN methodology is to establish a framework for comparing whether an intelligent management is comparable to the non-intelligent case, not only in terms of energy savings, but also in terms of throughput and hardware overheads. As such, a trivial case, where a simple threshold mechanism was used to determine whether or not a link would turn off or back on, was first implemented using an NoC simulation framework and the Orion power models [59]. The mechanism chooses an appropriate threshold based on which the links turn on and off. This trivial algorithm takes as input the link utilizations of all the links in the experimental NoC system, and outputs control signals based on a statically-defined threshold; based on this threshold, the algorithm then decides which links are turned off and then back on. The statically-defined threshold was computed based on simulation observations from different synthetic and realistic traffic models, and based on the observed power savings and throughput reduction when compared to a system without the mechanism. Figure 4.1 shows the real-time power savings for four synthetic traffic models and PARSEC traffic benchmarks, observed over a 4×4 NoC. (The reported result for PARSEC is the average across all the PARSEC benchmark applications).

This method was introduced in [32], and the results presented therein, as well

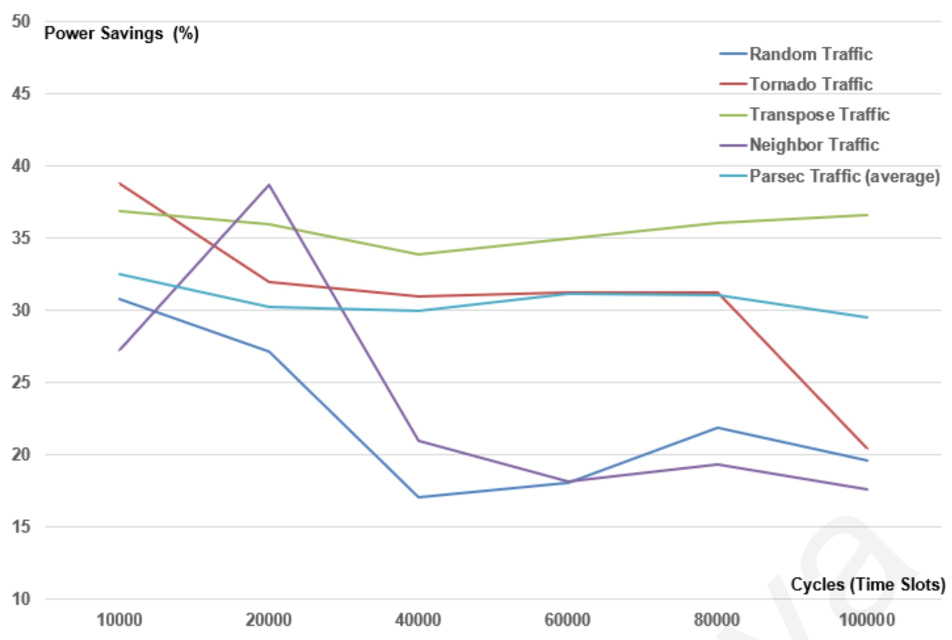


Figure 4.1: Power savings of a trivial threshold case compared to no on/off links case.

as the experiments with our framework indicate that such mechanisms can be quite effective. However, a run-time mechanism, which can be benefited from real-time information stemming from the network, can potentially outperform this method. Such mechanism is described next. Furthermore, [32] uses an open-loop mechanism, prone to oscillations that potentially can limit both the attainable performance and also the power savings, as power is still used during the transition [40].

4.3.2 Mechanism Overview

The ANN-based mechanism can be integrated as an independent processing element in the NoC, potentially located in a central point in the network for easy access by the rest of the PEs, and each base ANN mechanism can be assigned to monitor a NoC partition. Such cases are shown in Figure 4.2-(a). Each base ANN-mechanism monitors all the average link utilization rates within its region. These values are processed by the ANN, which computes the links that should be turned off and then back on, during each interval. Based on this, we turn off any links in the region that exhibit lower utilization. Links which have been turned off remain off for a certain period of time. Experiments in related work [32], [40] indicate that such time should be within a few hundred cycles, as longer periods tend to create a vast performance drop-off (as the network congestion increases due to lack of available paths), whereas shorter periods do not incur worthy power savings. The proposed

ANN mechanism uses a 100-cycle interval, during which all new utilization rates are received. This interval was chosen based on existing experiments in [32], which shows that a 100-cycle interval incurs better performance to power savings. The interval however, is a system parameter, which can also be taken into consideration by the system training, and involves future work which is presented in the next chapter. During the interval span, the ANN computes and outputs the links that should be turned off and then back on, which are then used by the link control mechanisms in each router to turn off underutilized links. The links remain off for another 100 cycles, and turn back on. During the 100-cycle interval, links which are off, do not participate in the computation of the next interval; instead, they are encoded with a sentinel value that represents them being fully utilized, so they are not kept off in two subsequent intervals. This reserves fair path allocation within the network.

Through this work, adaptive XY routing algorithm is used which can handle changes in the network topology. When links are turned off/on, the adaptive XY routing algorithm recomputes dynamically the available paths to reflect the changes. By rerouting packets using the available paths, the adaptive XY routing algorithm avoids turned off links. Thus, when links are turned off, reliable data delivery is guaranteed since adaptive XY routing algorithm dynamically adjust the routing based on the status of the network. Moreover, adaptive XY routing algorithm helps to avoid congestion by balancing the network traffic and provides deadlock free routing.

Each ANN-based mechanism follows a fully-connected multi-layer perceptron model [1], [2], consisting of one hidden layer of internal neurons/nodes and a single output-layer neuron. The activation function used in this work is the hyperbolic tangent function, which is symmetric and asymptotic, henceforth easy to implement in hardware as a LUT (Look-up Table) [32]. Furthermore, the specific function has been extensively used in several ANNs, and its accuracy has been great [1]. Back-propagation training algorithm is used for the training of the ANN. The ANN system is shown in Figure 4.2-(b). The number of internal neurons was chosen to be the half of the summation of the input and output neurons plus one [1]. The input neurons depend on the number of links that the system receives as feedback. As such, the size of the ANN depends on the number of inputs to the system. The output neuron chooses the corresponding links that best matches the pattern observed through the

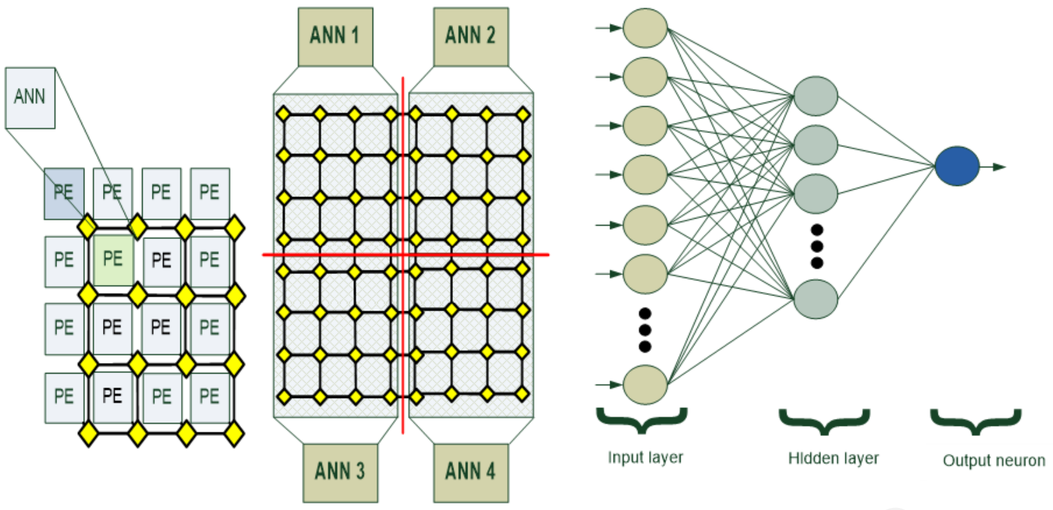


Figure 4.2: (a) ANN prediction with NoCs and an 8×8 network partition into four 4×4 networks with their ANNs, (b) Structure of the Neural Network.

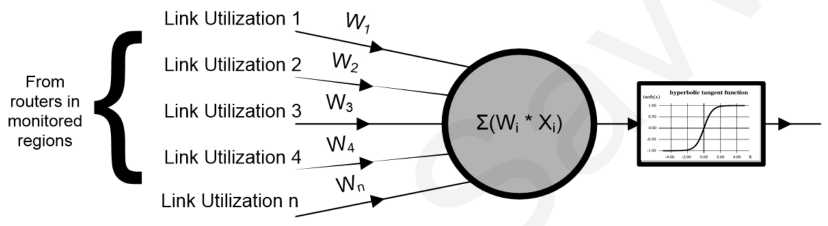


Figure 4.3: Neuron Computations.

hidden layer neurons, and outputs the underutilized links to the link controller.

The neuron computation involves computing the weighted sum of the link utilization inputs. An activation function is then applied to the weighted sum of the inputs of the neuron in order to produce the neuron output (i.e. activate the neuron). Equation 4.1 shows how the output of a neuron is calculated.

$$f(x) = K(\sum_i w_i g_i(x)) \tag{4.1}$$

K represents the activation function which is the hyperbolic tangent, w represents the weights which applies to the link utilization inputs which are represented by g(x) input function. The overall procedure is shown in Figure 4.3.

4.3.3 Intelligent Threshold Computation

While ANNs are heavily efficient in predicting scenarios based on learning algorithms, they require careful hardware design considerations, as their size and complexity depend on the number of inputs received, as well as the number of different output predictions (classes) that they have to do. NoCs consist of a large number of links which grows exponentially as the size of the NoC grows. Therefore, receiving link utilization and having to determine which links are candidates for turning off and then back on, would require an exponentially scalable ANN. As such, we devise a pre-processing technique, which identifies, based on simulation and observations, the set of candidate links for turning off and on, eliminating links which are almost always utilized. This depends obviously on the chosen network topology (for example, in a 2D mesh topology, links that are likely to be less busy include links which are at the edges of the mesh, whereas central links are usually more active and can be left on all the time), so that the ANN mechanism can handle the output decision in a more manageable way. Through various synthetic and realistic traffic simulations, for each given NoC topology, the average utilization values for each link through various phases in the simulation are computed, and the links with the highest utilization values, are always assumed that they will be on. Obviously this step reduces a little the effectiveness of the ANN, but it is necessary to minimize the size and overheads of the ANN both in terms of performance and in terms of hardware resources. This step has to be done for a given topology, prior to the ANN training. However, both steps (determining the links that the ANN will use, as well as the ANN training) can be done off-line, during the NoC design stage. The ANN training can also be done repeatedly whenever new application knowledge becomes available that might alter the on-chip network traffic behavior. This particular property of ANNs, provides a comparative advantage against a statically computed threshold, making the NoC flexible under any application that it is required to facilitate. It must be stated that the number of links that will be considered as likely candidates for on/off activity (i.e. the ones which do tend to have low utilization during the pre-training stage), impact both the size of the ANN itself, and the overall size of the mechanism (which involves logic that sends the appropriate control signals). Through the two steps, pre-training and training, each ANN can be trained and configured independently to satisfy its targeted NoC structure (topology and number of monitored links).

Furthermore, large NoCs can be partitioned into smaller regions. As such, a base ANN architecture can be assigned to monitor each region, and all the link utilizations of the routers of the NoC partition arrive at the ANN which is responsible for that region. The size of this NoC region, however, depends on two major factors; the incurred power savings that the corresponding base ANN offers, which depend

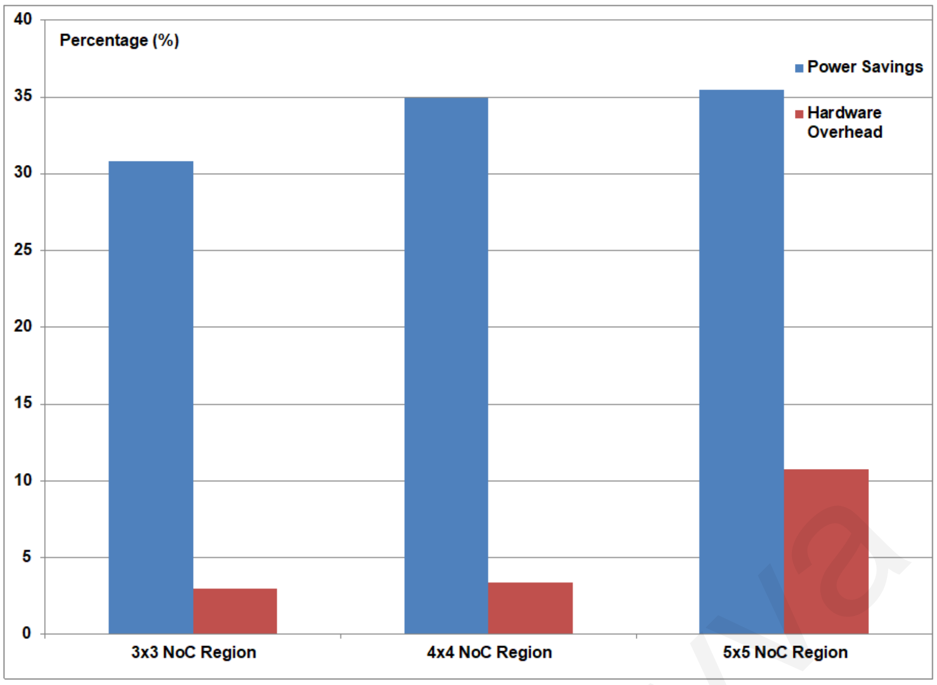


Figure 4.4: Power savings versus CMOS hardware overheads corresponding to various sizes of ANN monitoring regions in a NoC.

on its ability to process and evaluate the input information, and the resulting ANN size and hardware overheads (and subsequently power consumed within the ANN) which grow exponentially as the size of the NoC region grows. Choosing a small NoC region will likely result in a small ANN, but will result in smaller savings since the ANN will not have enough information to make accurate decision. On the other hand, a large NoC region will provide the ANN with much more information and potentially result in a much better decision, but its size and overheads would reduce the power savings making the ANN ineffective. As such, we experimented with several NoC regions and base ANNs, comparing their hardware overheads (a product of the ANN power consumption and the gate count required to implement each ANN in hardware) and responding savings incurred with the computed threshold. Figure 4.4 shows a comparison between hardware overheads (power $\times$ gate count) and power savings in the cases of 3×3, 4×4 and 5×5 ANN sizes for monitoring regions in an NoC. Results show that computation over a 4×4 NoC region offers satisfactory power savings and significantly less ANN overheads when compared to a 5×5 NoC region. A 3×3 NoC region does not provide enough information to the ANN in order to make accurate predictions. Based on these observations, we designed the base ANN system to monitor 4×4 NoC regions.

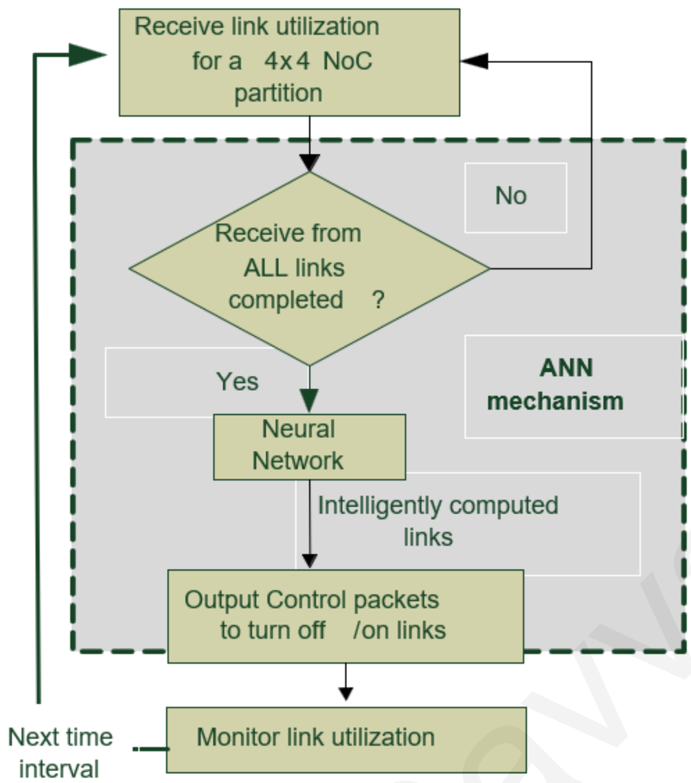

Receive link utilization for a 4x4 NoC partition
Receive from ALL links completed ?
No
Yes
ANN mechanism
Neural Network
Intelligently computed links
Output Control packets to turn off /on links
Monitor link utilization
Next time interval

Figure 4.5: Main steps of a 4×4 ANN predictor.

4.3.4 Base (4x4) ANN Operation and Hardware Architecture

The ANN mechanism is responsible to compute for all the link utilizations the minimum values during each interval. Based on these values, the ANN calculates the links to be turned off. Figure 4.5 shows the procedure of the ANN mechanism for a 4×4 NoC partition. The ANN mechanism receives all the average link utilizations from all the links of the 4×4 NoC partition. These values are fed to the ANN to be used for the prediction process. Each router contains a control hardware monitor that measures the average link utilization for each of the four links in each router, and this value is sent to the ANN every n cycles (where n is the size of the time interval). If a router fails to transmit the values at a single interval, its value is set to sentinel value, which shows that its buffers are fully utilized. This mechanism acts also as a congestion information mechanism because links which are heavily active are not candidates to be turned off. The ANN uses the utilization values to determine if a link is going to be turned off or on for the next n-cycle interval. As said earlier, we used 100-cycle intervals [32] (i.e. n = 100) in our simulations.

One of the main advantages of ANNs is their simple hardware implementation when the number of neurons remains small and the activation function remains simple [1]. The ANN hardware implementation depends on the number of hidden layer neurons. As mentioned in the Framework chapter, each neuron is implemented as a multiplier-accumulator (MAC) unit, with the accumulators being multiplexed for each neuron, so that the number of multipliers is minimized. The base ANN hardware architecture is shown in Figure 4.6. Utilization values for each link arrive and sorted through an input coordination unit, which distributes the values to each of the appropriate multipliers. The multipliers receive these values and through a shared weights memory, receives the corresponding weight. The weights and inputs product is then accumulated in the corresponding accumulator, with the entire process controlled via a finite-state machine controller. Each neuron has an assigned storage register, to enable data reuse; when one layer of neurons is computed, their outputs are stored inside a corresponding register. As such, the same hardware is reused for computing the next layer (i.e. from input layer to hidden layer, and from hidden layer to output layer). When each neuron finishes its MAC computation, the result is then computed through the activation function LUT, and propagates to the output neuron.

An ANN monitoring a 4×4 region in a mesh topology for example, receives 64 different inputs; if we are to assume that each router transmits a packet with its own link utilization during each interval, and if we also assume one packet per cycle delivered to the ANN during each interval, then, during each cycle, the ANN will receive at most 4 input values. Hence, if we use pipelined multipliers, we need only 4 multipliers for each ANN to achieve maximum throughput. The ANN therefore remains small and flexible, regardless of the size of the network it monitors. Furthermore, an ANN monitoring a 4×4 NoC partition, receives 16 packets (one for each router); as such, it requires 16m cycles (where m is the cycle delay of each multiplier), plus 16 cycles for each accumulator, plus one cycle for the activation function plus one cycle for the output neuron, to output the new threshold (total of 16m+18 cycles). The overall data flow and architecture is shown in Figure 4.6.

4.4 Hardware Optimizations and Tradeoffs

In order to make the ANN architecture simpler and smaller we studied how the number of neurons of the hidden layer affect the total power savings of the system. Given that the 4x4 ANN monitors sixteen routers, we need sixteen input neurons

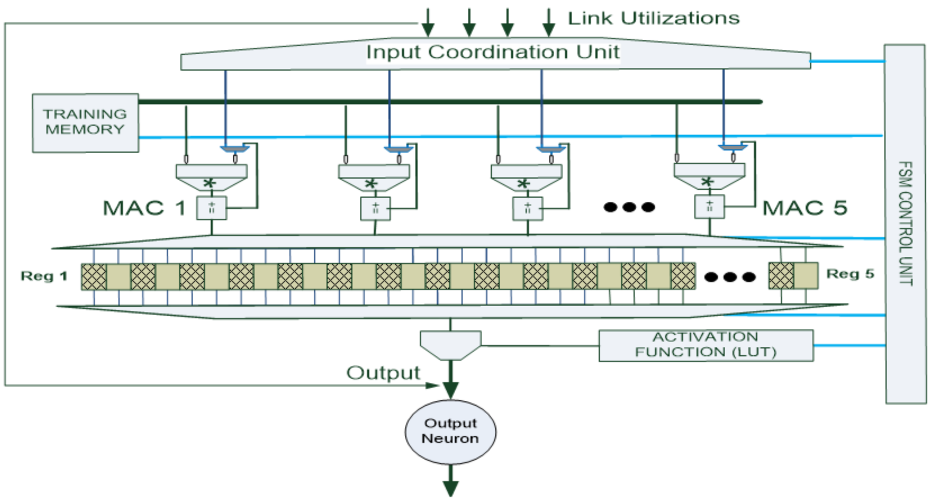


Figure 4.6: ANN hardware architecture and its hardware realizations.

[2]. Having sixteen neurons at the input layer of the ANN means that the hidden layer should have seventeen neurons (based on the rule of thumb that a satisfactory number of the hidden layer neurons equals to half the number of input and output neurons plus one neuron) [2]. Three different ANNs were developed with seventeen, sixteen and fifteen neurons at the hidden layer respectively. Figure 4.7 shows the power savings for these ANNs under the use of four different traffic patterns (Random, Tornado, Transpose, Neighbor) and data from PARSEC traffic benchmarks. Using sixteen neurons therefore (instead of seventeen), in the hidden layer exhibits the best power savings for all the traffic patterns. (The reported result for PARSEC is the average across all the PARSEC benchmark applications). In addition, we studied how the bit representation of the training weights affects the threshold computation and subsequently the total power savings. Figure 4.8 shows how the bits used in representing the training weights influence the power savings of the system. As we can see 24, 16, 8 and 6 bits show similar power savings, but these savings are significantly reduced when 4 bits are used, due to reduced training accuracy. Based on the above, we selected the weight bit representation to be 6 bits, which made the multiplier-accumulation hardware very small, requiring a 6-bit port for each weight and a 5 bit port for the utilization values. (The reported result for PARSEC is the average across all the PARSEC benchmark applications).

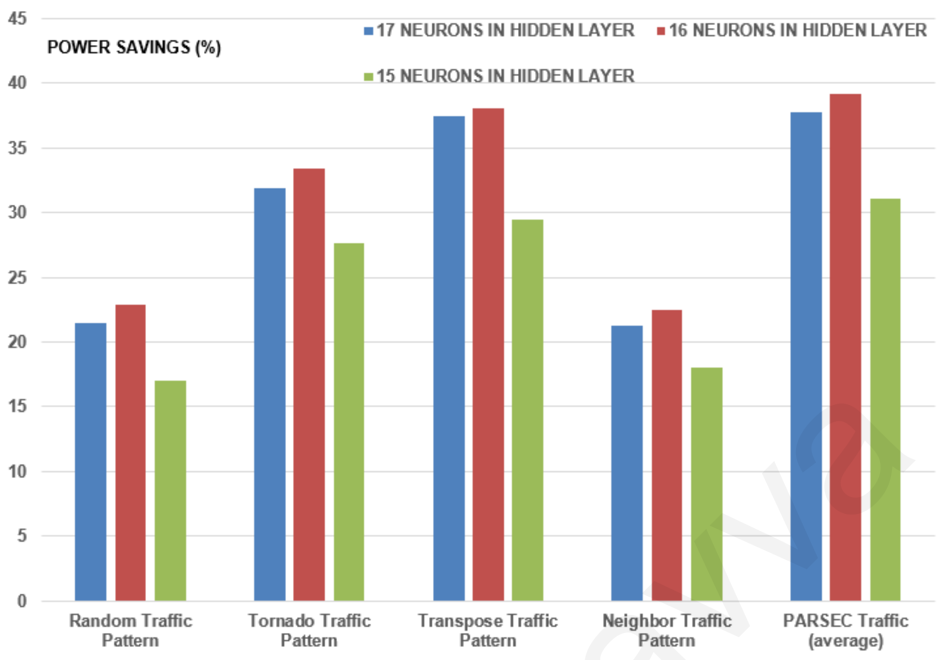


Figure 4.7: Power savings for 17, 16 and 15 neurons in the hidden layer of the ANN.

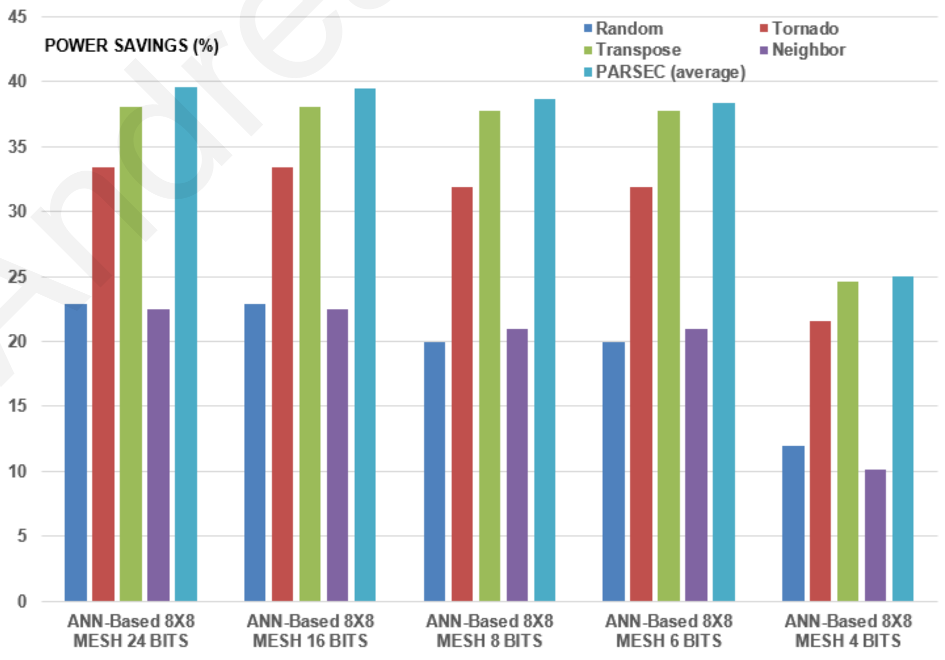


Figure 4.8: Power savings for different training weight bit representations.

4.5 Simulations and Results

In order to evaluate the ANN-based on/off link prediction mechanism, gpNoCsim simulator is used as previously mentioned [58]. We experimented with 8×8 mesh and torus topologies. Simulations are done over a range of 1,000,000 cycles, with a warm-up period of 100,000 cycles. In the 8×8 topologies, we partitioned the NoC into four regions of 4x4 routers/links, where each ANN-based model was assigned as responsible for monitoring. The ANN-based models monitored all links in their corresponding partition, and all links were candidates for off/on, and all ANN results related to the size and operation of the ANN are given based on these architectural details.

Time was divided into 100-cycle intervals [32]; at the end of each interval, all routers in the NoC partition transmit their average utilization data for that span. A time-out mechanism equal to the expected delay of each router towards the ANN mechanism is imposed, to maintain reasonable delays. The ANN receives one packet from each router with four utilization values, one for each port. The ANN then proceeds to the prediction which is transmitted to each router through a control packet. Each router then turns off each link, depending on the intelligent prediction. The router continues operation until the end of the new interval. It must be repeated that when a link is turned off or on, an extra 100-cycle penalty is inserted into the simulation, to indicate the impact on the network throughput.

In order to study the power savings and the throughput of the dynamic ANN-based prediction algorithm for turning links on/off we compare this to a static threshold-based algorithm and to a system without any on/off mechanism. Using synthetic and realistic traffic patterns (Random, Tornado, Transpose and Neighbor) and data from PARSEC benchmark workloads with varied injection rates [60], [63], we first evaluated the power savings of the ANN-based mechanism when compared to the same system without any on/off link capability, and when compared to a system that employs a statically determined threshold. The traffic patterns, for which we experimented, are a superset of the patterns used to train the ANN; we measured power savings and the impact of the throughput on all the traffic patterns however. In order to compute the power savings in the torus network, we follow the guided-training approach as described previously, and we measure link utilizations in all possible partitions of the torus network to compensate for the toroidal

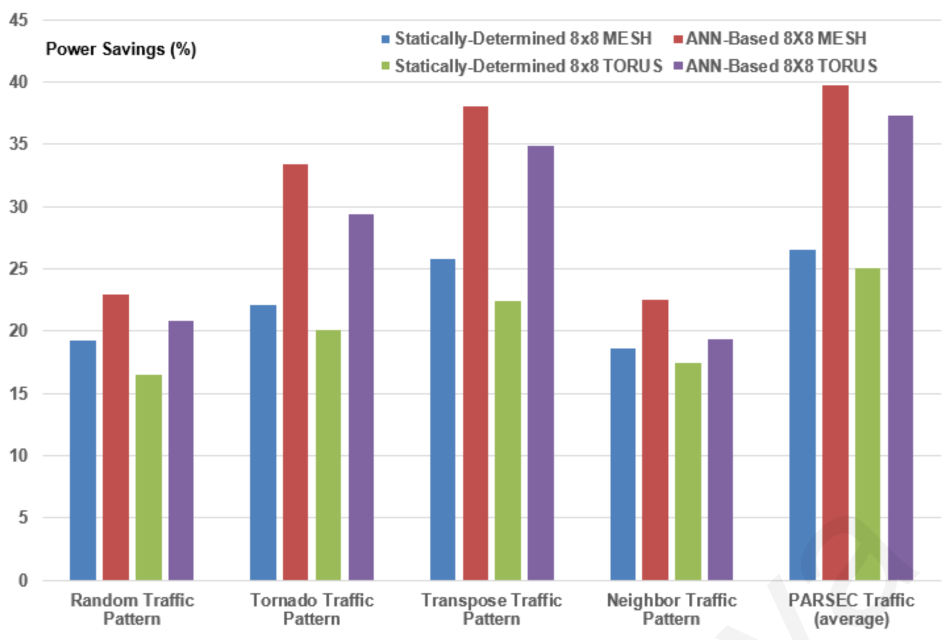


Figure 4.9: Power savings for 8x8 mesh and 8x8 torus networks for the ANN based technique, static threshold technique and no on/off technique.

links. The link utilizations with the least values (from all the link utilizations, from all the partitions of the torus network) are then passed through the ANNs. Figure 4.9 shows the comparison when targeting 8x8 mesh and torus NoCs. The power savings of the ANN-based mechanism are better than the savings in the cases of statically determined threshold and the case without any on/off links. Results show that the ANN-based mechanism can achieve 13% additional power savings when compared to a statically-determined threshold methodology. (The reported result for PARSEC is the average across all the PARSEC benchmark applications). The ANN-based mechanism can identify a significant amount of future behavior in the observed traffic patterns; therefore, it can intelligently select the underutilized links for the next timing interval.

Next, we measure the impact of the throughput in each mechanism; while having no on/off mechanism obviously yields a higher throughput, the ANN-based technique shows better throughput results compared to statically determined threshold techniques. Figure 4.10 shows the throughput comparisons for an 8×8 mesh and an 8×8 torus network. The ANN-based methodology shows around 6% better throughput results when compared to statically based methodology. The throughput values are normalized based on the number of the simulation cycles. (The reported result for PARSEC is the average across all the PARSEC benchmark applications).

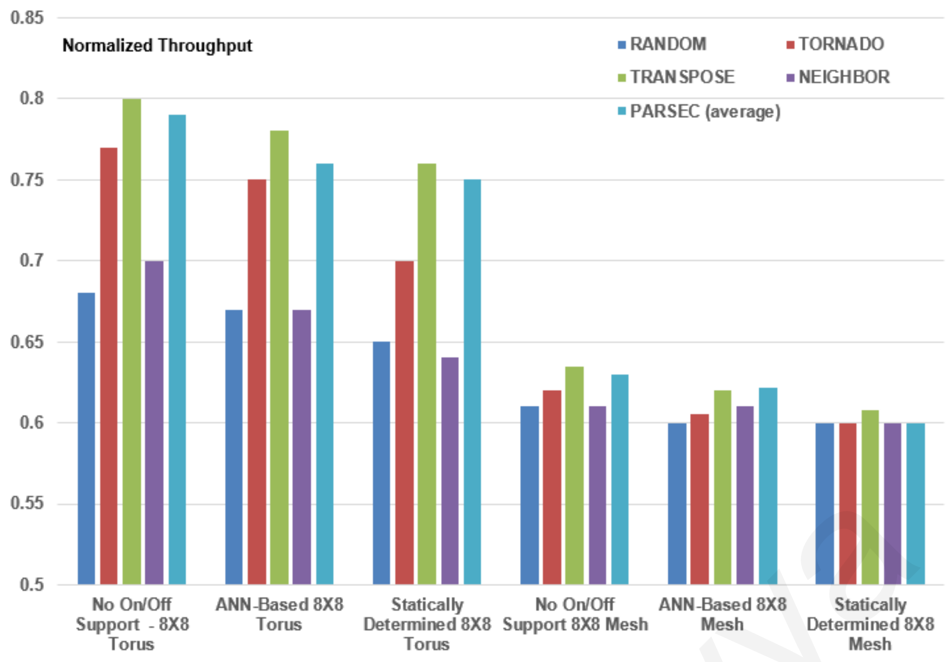


Figure 4.10: Average network throughput comparisons for 8x8 mesh and torus networks.

Figure 4.11 represents the normalized energy consumed in an 8×8 torus network. We observe that the energy consumed using the ANN mechanism is less than the cases of statically-computed threshold and without on/off link management algorithm. The ANN exhibits a reduction in the overall energy, because of a balanced performance-to-power savings ratio, when compared to not having on/off links or when compared to static threshold computation.

Figure 4.12 presents the average packet delay in packets per cycle for the 8x8 mesh, when the ANN-based mechanism is used compared to the cases where no on/off mechanism is used and the statically computed threshold case. The ANN-based mechanism incurs more delay (less than 6% additional packet delay), but we believe that the delay penalty is acceptable when compared to the associated power savings. (The reported result for PARSEC is the average across all the PARSEC benchmark applications).

4.6 ANN Hardware Overheads - Synthesis Results

To compute the hardware overheads of the proposed scheme, the ANN-based mechanism for one 4×4 NoC region, was synthesized and implemented targeting a commercial 65nm CMOS technology. The ensuing synthesized ANN-based controller

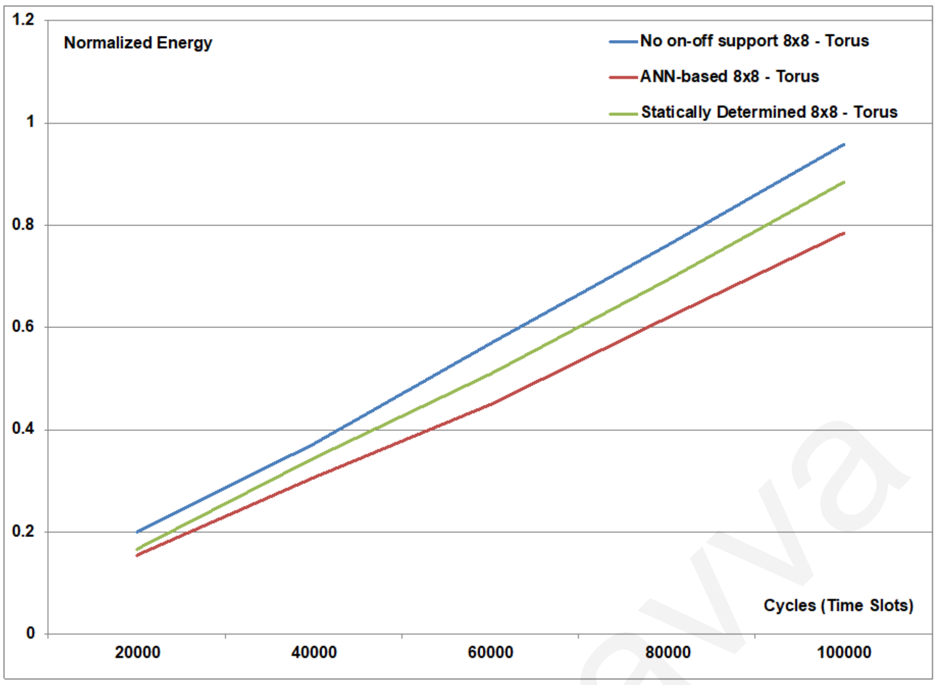


Figure 4.11: Energy consumption for an 8x8 network.

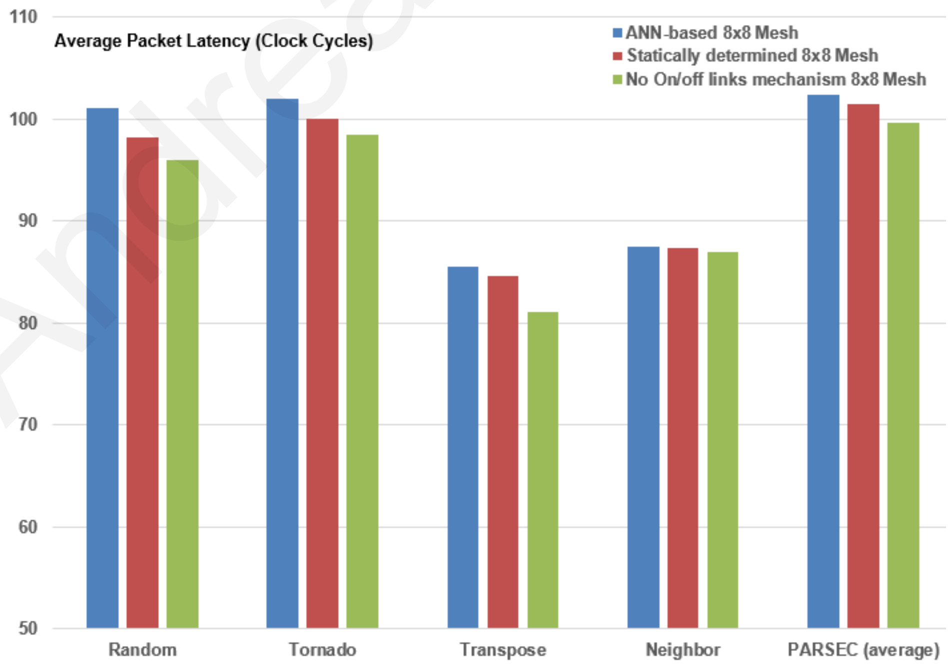


Figure 4.12: Average packet latency for the cases where ANN-based mechanism is used, when trivial case is used and when there is no on/off mechanism.

and the associated hardware overheads in each router, consume approximately 4 K logic gates (for comparison purposes, an NoC router similar to the one used in our simulation [64] consumes roughly 21K gates), bringing the estimated hardware overhead for an 4×4 mesh network to roughly 4% of the NoC hardware.

Assuming 50% switching activity probability, the synthesized ANN described consumes an estimated 0,0022mW when computing one cycle of 16 inputs (one full router utilization packet). A total of 16 cycles is needed for multiply – accumulate, 1 cycle for the thresholding and 1 cycle for the activation function. Assuming that the ANN can start receiving data in Cycle 1 and with a steady flow of one utilization packet from each router, then the inputs need 19 cycles in order to reach the next layer of neurons. So 19 cycles x 0,0022mW equals 0,0418mW in total, in order to reach the next layer of neurons. The next stage will use less power and the same hardware can be reused so there will be no need to compensate for extra area.

4.7 Comparison with Related Works

Lastly, we briefly give a comparison with relevant related works that follow dynamic threshold techniques in Table 4.1.

In [30], authors use a DVFS technique for power optimization of interconnected networks based on a history-based DVS policy. This policy realizes up to 30% power savings with an increase of 15,2% in the average latency. Additionally, based on synthesis with Synopsis Design Compiler this method presents hardware overheads – around 500 logic gates per router port. In [32], authors propose a dynamic power management policy for turning links off/on based on the derivation of a connectivity graph that balances power and performance and based on a fully adaptive deadlock free algorithm. This method realizes up to 37,5% reduction in the overall power and an increase in latency of the scale 48,5% (- no hardware overheads are presented). In [65], authors propose LEAD (Learning-Enabled Energy-Aware Dynamic Voltage/Frequency Scaling), which includes a collection of linear regression DVFS techniques that are all trained offline. This method realizes an average of 20% power savings with no latency increase. This method presents additional overheads for the on-chip voltage regulators caused from the addition of switching components and for the feature set used which includes 39 network parameters. When compared to all [30], [32] and [65], the ANN-based method yields better power savings

Table 4.1: Power savings/hardware overhead comparisons.

Related Work	**Characteristics**	**Power Savings**	**Hardware Overhead**
[32]	8x8 2-D mesh topology, Uniform traffic	Around 37,5% - turning on/off links	N/A
[30]	8x8 2-D mesh topology, Pareto distribution – 0.5 packet injection rate	Around 30% (DVFS technique)	500 equivalent logic gates per router port
[65]	4x4 mesh topology, PARSEC benchmarks	Around 20% average savings (collection of linear regression based DVFS)	Additional overheads for regulators and for the feature set
Proposed ANN technique	8x8 2-D mesh topology and 8x8 torus topology Uniform traffic	Up to 40% - turning on/off links based on ANN prediction	4% of the NoC hardware for a complete 4×4 Mesh NoC

(around 40%), while still maintaining minimum hardware overheads. Additionally, our method presents better average latency results when compared to both [30], [32] works. Our method is easier to implement in comparison with past methods since the hardware overheads are kept to minimum levels. We must note that while [40] was the motivating idea behind our work, it presented only a preliminary implementation of the idea, without enough information about hardware overheads and power savings in order to make an informed comparison. Our method outperforms this work since we use real-time information for training. Our work provides simulations and evaluations of different important parameters in order to achieve optimal results. Moreover, in our method, power savings are achieved through the use of small and scalable ANNs. This gives an advantage against statical methods making our framework easy to adjust to any application that is required to accommodate. Additionally, by switching off/on links instead of using Dynamic Voltage and Frequency Scaling techniques our work provides better power savings advantages avoiding static energy power consumption.

4.8 Conclusion

This chapter presented how an ANN-based mechanism can be used to dynamically compute, based on appropriate training, candidate links for turning off and then back on, in an effort to achieve power savings in an NoC. The ANN-based model utilizes very low hardware resources, and can be integrated in large mesh and torus NoCs, exhibiting significant power savings. Simulation results indicate approximately 13% additional power savings when compared to a statically-determined threshold methodology under synthetic and realistic traffic models.

Chapter 5

ANN-Based Fault Detection in Hardware Systems / NoCs

5.1 Introduction

ANNs can be used for fault detection purposes since they have excellent characteristics such as generalization capability, robustness and fault tolerance. Designing an ANN in order to be used for fault detection purposes includes different parameters. Through this chapter, those parameters are presented and analyzed based on simulations. Moreover, after the development of the ANN, in order to evaluate it, a case study scenario based on Networks on Chip is used for detection of inter router link faults. Simulation results with various synthetic and realistic traffic models are implemented and analyzed.

This chapter provides analysis and evaluation of the procedure of creating an exploration framework with the use of ANNs for high level fault detection in hardware systems. While implementing a detection framework, there are many issues to deal with, which have inspired researcher's attention. Determine the ANN network structure, the number of neurons in hidden/output layers and the procedure of the ANN training for fault detection are some of those issues. The aim of this work is to identify and analyze all the main steps needed for this purpose.

Through this work, all the necessary steps for designing such a mechanism are analyzed, explained and evaluated. The developed ANN is adaptable in many different hardware systems. Based on their ability to dynamically be trained to different case scenarios, ANNs can offer high detection capabilities with minimal additional overheads [1]. An ANN mechanism for fault detection is developed in order to intelligently detect future faults. The ANN is trained with utilization data collected from different simulations in which randomly faults were injected (topology based training). Added to this, individual small ANNs are assigned to be responsible for different hardware partitions, providing scalability. ANNs can easily be designed and implemented in hardware and their size for this work remains relatively small.

5.2 ANN Mechanism Overview

Nowadays, the complexity and density of hardware increases the need for intelligent fault detection mechanisms. The relation between utilization and fault detection can be complicate. Likelihood of faults can be increased since high utilization can put more tension on the hardware. Also, unusual variations in the link utilization rates may be a sign of hardware issues. Based on this, it's important to achieve a balance between fault detection and utilization. Utilization must be taken into account for effective fault detection methods. Our work, based on real-time data and based on the use of ANNs, can intelligently predict future faults by taking into consideration the utilization of the system. Additionally, our work can help maintaining the utilization high by identifying future faults and reducing their influence on the system performance.

The ANN mechanism for our work can be considered as an independent processing element in the NoC. Each base ANN mechanism is responsible for a specific network partition. The ANN mechanism monitors all the link utilization values in the region that is responsible for, and these values are then processed by the ANN in order to make the detection. Link faults are considered in this work, which involve faults on the links connecting the routers between them and the processing elements on the NoC topology. Injecting link faults can represent various types of link fault scenarios like link failures due to physical damage, circuit faults or manufacturing defects.

Every x cycles, new link utilization values are coming to the ANNs for training. The ANN then, based on the training phase, intelligently detects which routers will be erroneous for each random fault injection. An overview of the procedure that an ANN mechanism follows in order to detect which routers will be malfunctioning is

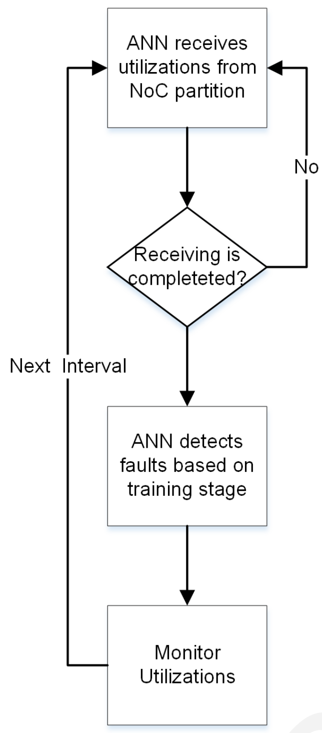

ANN receives utilizations from NoC partition
No
Receiving is completeted?
Next Interval
ANN detects faults based on training stage
Monitor Utilizations

Figure 5.1: Overview of the ANN detection process.

presented in Figure 5.1.

Next sections present in detail the development of the framework based on the previously mentioned steps. For each decision step, an 8x8 NoC case-study is going to be simulated with various traffic patterns in order to evaluate each decision's output.

5.2.1 Scalability - ANN partitioning

For this work, partitioning of the NoC case study into smaller parts (e.x. an 8x8 NoC into four 4x4 regions or four 4x5 regions or 5x4 regions) is needed and an ANN is assigned to be responsible for each part. The partition will help to keep the ANN sizes small with minimum complexity.

In order to decide which region size is the best, different partitions are created and compared. Different simulations for the different partitions are created and the resulting Receiver Operating Characteristics (ROC) curves are studied. Based on those plots an appropriate ANN topology is chosen. Figure 5.2 shows the different NoC partitions studied (4x4, 5x4, 4x5).

Figure 5.3 presents the resulting ROC curves for 4x4, 4x5 and 5x4 topologies in the case of router fault detection. Different ROC curves were created with the

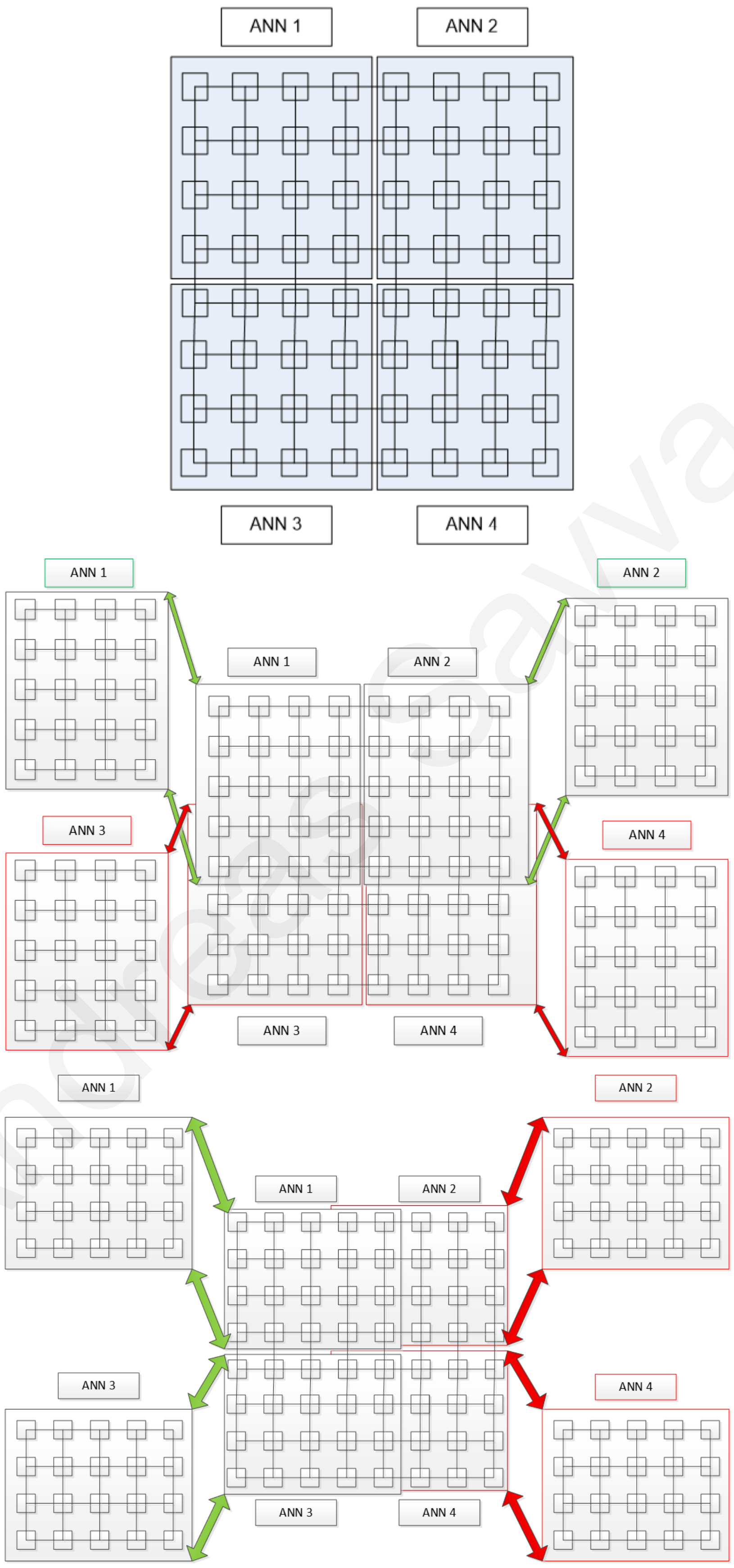
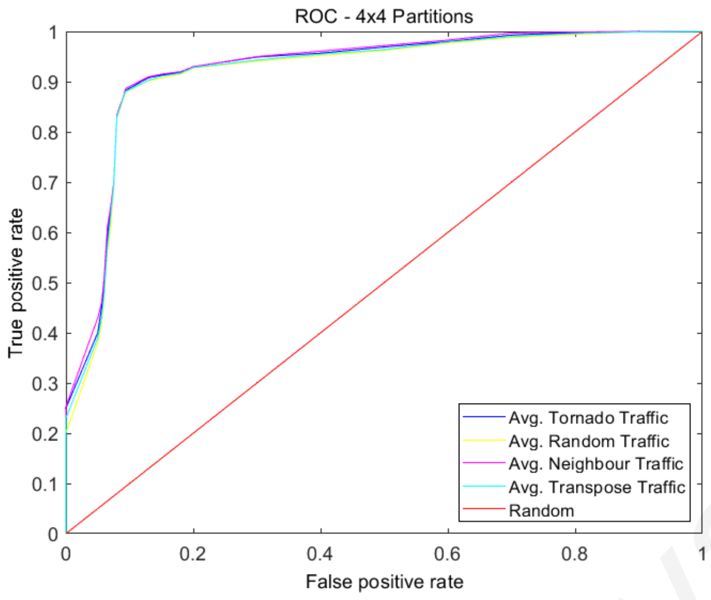
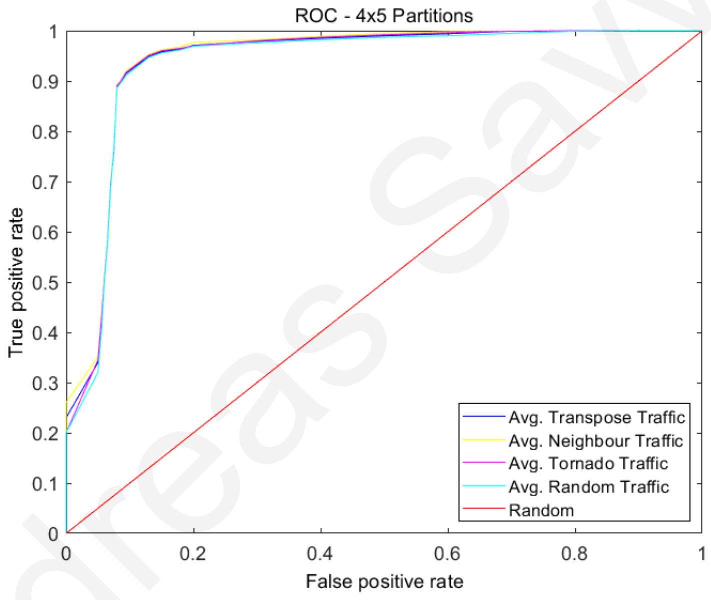


Figure 5.2: Different NoC partitions: 4x4, 5x4 and 4x5 partition.





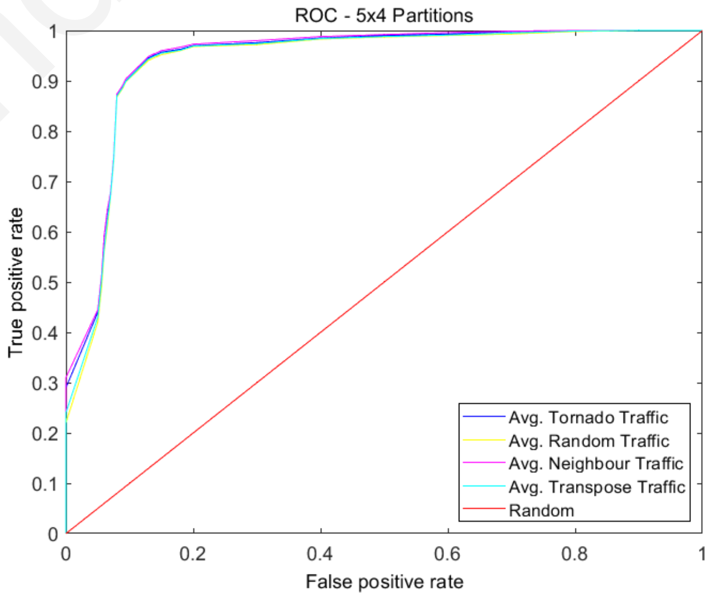


Figure 5.3: Resulting RoC curves for 4x4, 4x5 and 5x4 partitions.

use of different traffic patterns. Partition 4x5 is more efficient since it produces better resulting ROC curves for different traffic patterns compared with 4x4 and 5x4 partitions (- partition 4x5 ROC curves present slightly better results compared with the results of the 5x4 partition. This is more obvious near the 0.9 value of the true positive rate in both ROC graphs). Based on the above, we chose to work with 4x5 partitions, for the case of detecting which routers in the region will present fault.

5.2.2 ANN Development for Fault Detection

Each ANN follows a fully connected perceptron model. The activation function used is hyperbolic tangent, which is symmetric and asymptotic as mentioned in the previous chapter. This makes it easy to implement in hardware [22]. The neuron computes the weighted sum of the utilization inputs and then through the activation function the neuron output is produced.

The ANN is trained, based on different traffic models (Random, Tornado, Transpose, Neighbor and realistic traffic patterns based on the use of data from PARSEC benchmark suite), using an off-line and back – propagation ANN training algorithm. One hidden layer is used for the ANN since more hidden layers rarely improve the system and can converge to local minimum [22]. For the purposes of this work, the Matlab ANN toolbox is also used along with the different traffic patterns.

The input neurons were chosen based on the number of the inputs to the system. For the output neurons, two scenarios were studied for this work. In the first scenario, the ANN is responsible to detect which partition will present a fault. For the second scenario, the ANN is responsible to detect which routers in the partition will present fault. For these partitions, in the case of detecting fault in the whole NoC partition, the ANN should only have one output neuron. For the case of detecting which routers in the region will more likely present fault, the ANN should have 20 output neurons, one for each region router in the cases of 5x4, 4x5 NoC partitions and 16 output neurons for the 4x4 NoC partition case.

Figure 5.4 and figure 5.5, present the ANN architecture used in the two different case scenarios for this work. Figure 5.4 shows the ANN architecture for the case of detecting the partition which will have a faulty router. Figure 5.5 shows the ANN architecture for the case of detecting which routers in the NoC region will present fault. The difference is presented in the output layer. In the first case, only one

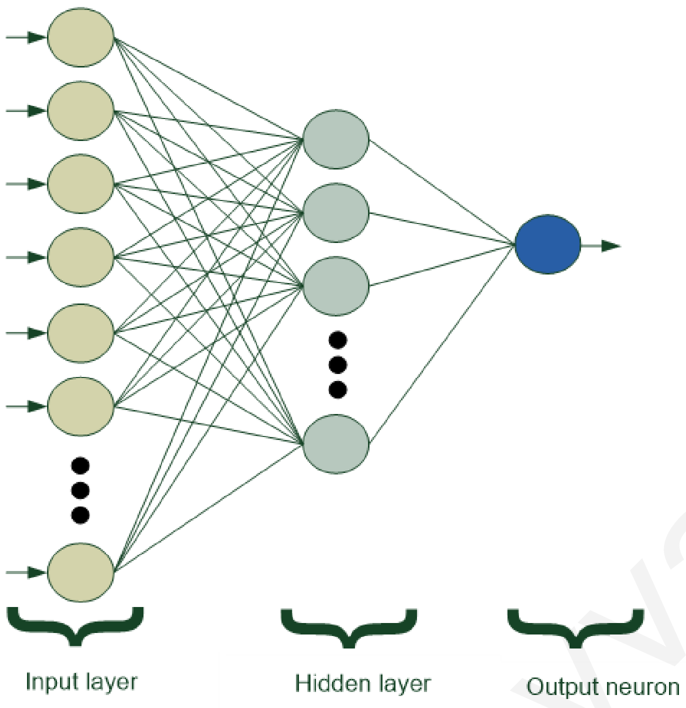


Figure 5.4: ANN architecture with the input, hidden and output layers for detection of fault in whole ANN.

output neuron is needed which will show which NoC region might have fault. In the second case, the number of output neurons depends on the number of the routers included in the NoC region which is simulated.

5.2.3 (4x5) ANN Based Model

As mentioned in the previous chapter, an ANN which is responsible to monitor a 4x5 region receives 80 different inputs under the assumptions that each router transmits a packet with its own link utilization and one packet per cycle is delivered to the ANN at each interval. Based on that, during each cycle the ANN will receive for each router at most four input values. Only four pipelined multipliers are needed for each ANN and the ANN remains small and flexible and independent of the size of the NoC it monitors. The ANN hardware architecture and the overall data flow are shown in the previous chapter.

5.2.4 ANNs Parameters and Training

The ANN mechanism for this work operates in two steps, training step and detection step. In order to train the ANNs correctly, the utilization values collected from the

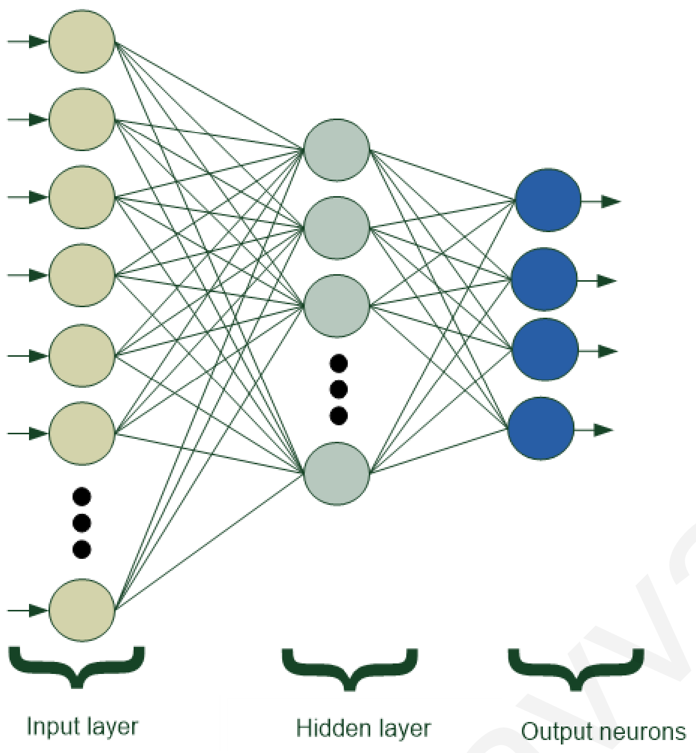


Figure 5.5: ANN architecture with the input, hidden and output layers for detection of fault in router.

NoC simulation are used. The ANN receives the link utilization values of all the router ports of the partition that monitors. The ANN then uses these utilization values for training in order to intelligently detect which routers will present fault.

The ANN training stage can be performed off-line when the NoC is not used and the training weights can be stored in SRAM based LUTs for fast and on-line reconfiguration of the network, as mentioned in the Framework chapter. The network is trained off-line, with the use of different application traffic patterns. In our experiments we used synthetic and realistic traffic patterns (Random, Tornado, Transpose and data from PARSEC benchmark suite) and the Matlab ANN toolbox.

For the purposes of our work, the training sets are not used for evaluation purposes, ensuring that our method has accurate results - training sets and testing sets used are mutually exclusive. Back-propagation algorithm is used for training the neural networks in order to correctly adjust the neural network's parameters. Back-propagation updates the weights of the ANNs to minimize the loss.

Figure 5.6 shows the training, validation and testing results. The dashed line shows the perfect result and the solid line shows the best fit between the outputs and targets. The R value represents the relationship between the outputs and targets.

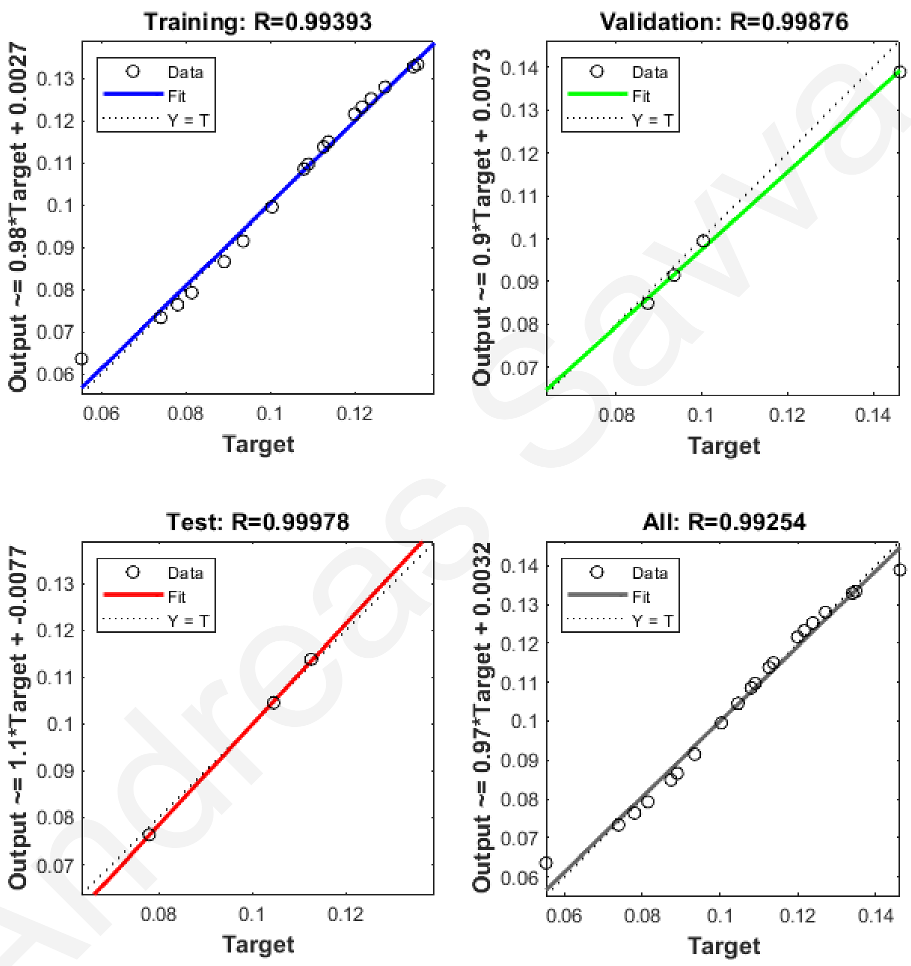


Figure 5.6: Training, Validation and Testing results.

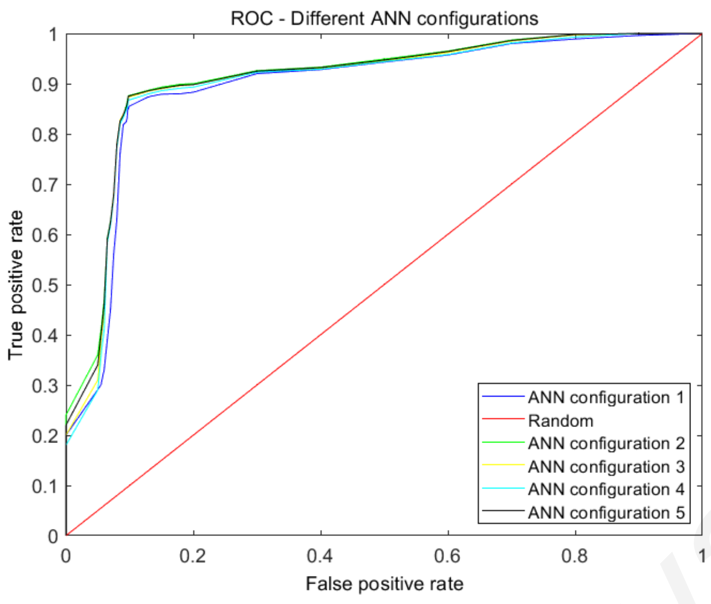


Figure 5.7: RoC graphs for different neurons in hidden layer.

Table 5.1: ANN configurations with different number of neurons in the hidden layer.

ANN Configuration	**Number of Neurons in the Hidden Layer**
ANN Configuration 1:	17 Neurons in the Hidden Layer
ANN Configuration 2:	21 Neurons in the Hidden Layer
ANN Configuration 3:	20 Neurons in the Hidden Layer
ANN Configuration 4:	18 Neurons in the Hidden Layer
ANN Configuration 5:	19 Neurons in the Hidden Layer

If R is equal to 1, this shows that there is an exact linear relationship between the outputs and targets. If R is near zero then there is no linear relationship between outputs and targets. Based on our results, the data shows good fit. The validation and test results have the R near 1 which shows that there are no significant issues with the training, validation and testing.

The next decision step presented is the number of neurons needed for the hidden layer. In order to minimize the fault probability and have a well-trained neural network which will perform well, an optimal value for the neurons of the hidden layer must be selected. If a small number of neurons are selected, it will lead to faults for the total framework as the training data might not be well used for detection within that small number of internal neurons. If a huge number of neurons are selected for the hidden layer, then this will add extra hardware implementation cost

Table 5.2: ANN configurations for different sentinel values.

ANN Configuration	**Number of Neurons in the Hidden Layer/Sentinel Value**
ANN Configuration 1:	19 Neurons in the Hidden Layer, Sentinel value 70
ANN Configuration 2:	19 Neurons in the Hidden Layer, Sentinel value 50
ANN Configuration 3:	19 Neurons in the Hidden Layer, Sentinel value 20

and faults. The number of hidden layer neurons can be selected to be half of the summation of the input and output data plus one [1]. Based on these, different numbers of internal neurons, varying around half of the summation of the input and output neurons, are studied and simulated.

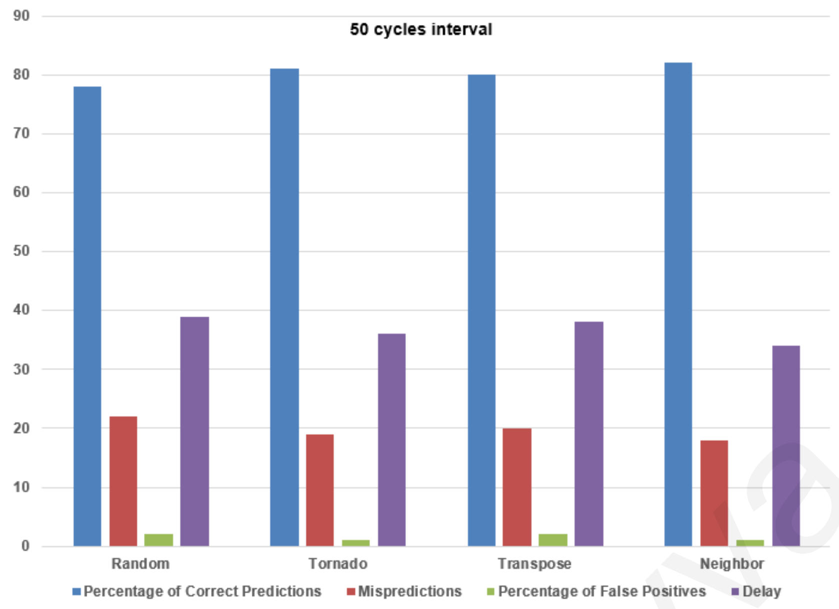
Figure 5.7, shows the resulting ROC graphs for different number of neurons in the hidden layer. Table 5.1 presents the configuration parameters used for each resulting ROC graph.

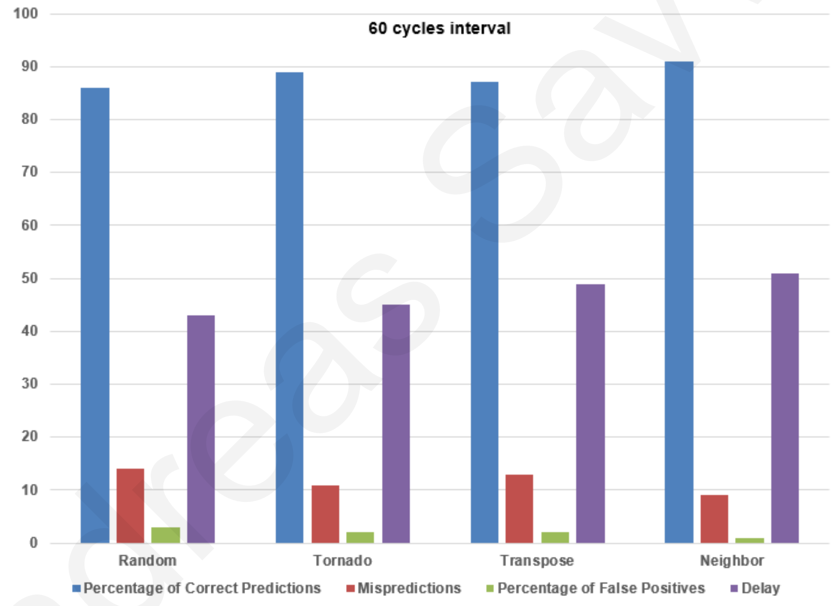
Based on the figure above, the configuration with 19 hidden layer neurons is chosen. This is because ROC curve for 19 neurons in hidden layer shows good results compared to the rest of the cases.

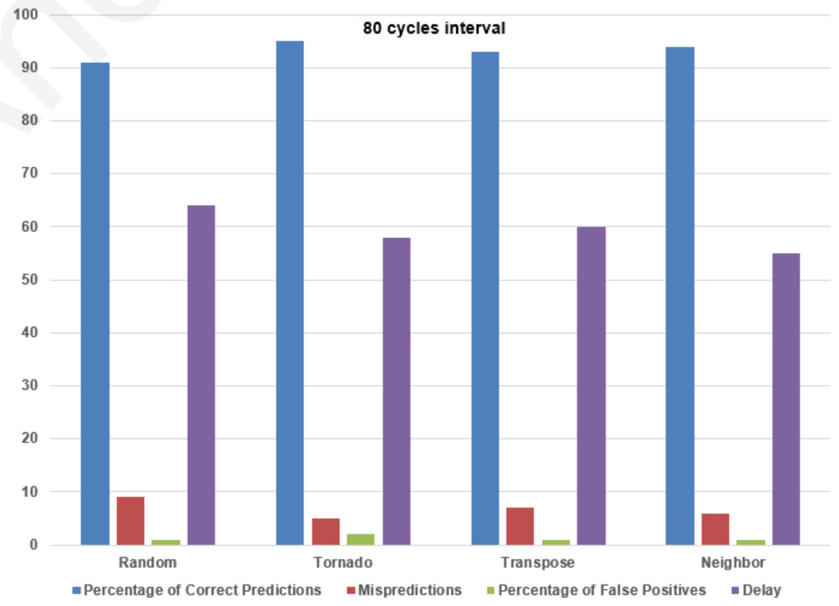
5.2.5 Simulation Decisions for Fault Detection

Another important decision that needs to be examined is the sampling period. Sampling time is divided into different cycle intervals and the experimental results were studied. At the end of each interval, all routers in the partition transmit their average utilization data. The ANN then receives the utilization values and proceeds to the detection. If a small sampling period is chosen, then the utilization data which will be collected will not be sufficient for the detection. 50, 60, 80, 100 cycle time intervals were studied and based on the experimental results an optimal time interval is selected. Based on the results of Figure 5.8, the 80 cycles time interval is chosen because it shows better detection results compared with the rest of the cases.

In order to find a good sentinel value, which will be used when a router fails to transmit its values to the ANN, different simulations for different sentinel values are created. Table 5.2 presents the configuration parameters used for each resulting ROC graph in figure 5.9.


50 cycles interval
90
80
70
60
50
40
30
20
10
0
Random
Tornado
Transpose
Neighbor
Percentage of Correct Predictions
Mispredictions
Percentage of False Positives
Delay


60 cycles interval
100
90
80
70
60
50
40
30
20
10
0
Random
Tornado
Transpose
Neighbor
Percentage of Correct Predictions
Mispredictions
Percentage of False Positives
Delay


80 cycles interval
100
90
80
70
60
50
40
30
20
10
0
Random
Tornado
Transpose
Neighbor
Percentage of Correct Predictions
Mispredictions
Percentage of False Positives
Delay

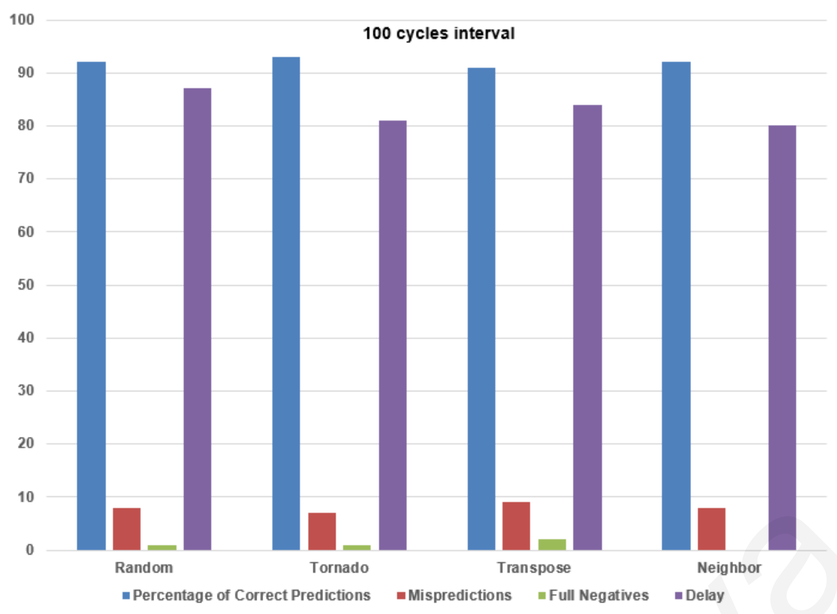


Figure 5.8: Results for different cycle time intervals.

Figure 5.9, present the resulting curves. The best results comparing the ROC curves for the different sentinel values are shown in the case of 50 cycles sentinel value. Based on the results, an efficient value to work with is 50 cycles.

5.2.6 Topology Exploration Setup – Adaptability in Various Hardware Configurations

Simulation experiments were ran over 8x8 NoC topologies, case study scenario, where partitions into four regions of 4x5 routers are created. Each ANN based model is responsible for one of these regions. Simulations are done over 1,000,000 cycles with warm-up period of 100,000 cycles. Time is divided into 80 cycles (sample period), as already explained, and at the end of each sampling period all the routers of the partition send their average utilization data to the ANN which is responsible for each NoC partition. Faults are injected randomly (random cycle time and in random locations). The ANN then receives the average utilization values (one packet from each router with the average utilization port values of this router). The ANN then proceeds to the detection of faults.

One simulation runs at a single traffic pattern, with a single traffic injection rate and one network state consists of 223 different simulations (for all the injection faults in all the 223 links – single fault injection is assumed.) Traffic injection rate is varied from 0.1 – 0.3 flits/node/cycle in steps of 0.05 flits/node/cycle from low to

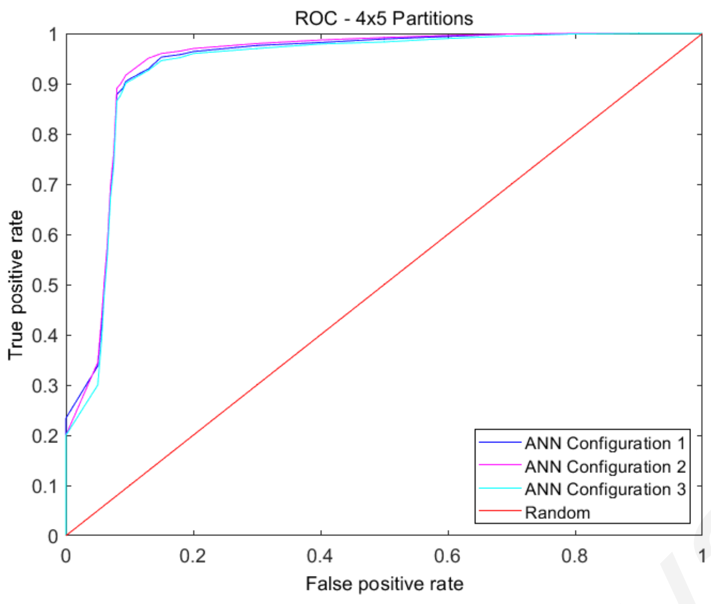


Figure 5.9: Resulting RoC graphs for different sentinel values.

high. Added to this, different traffic patterns are studied in three different scenarios of fault injections (at cycles: 0, 32000 and 64000). Three injection rates multiply three injection times multiply different traffic patterns multiply 223 links equals more than 8000 fault injection simulations. Simulation results are presented next.

The number of correct detections is measured as well as the number of fault positives (undetected faults) and the number of full negatives (unexpected faults) for different traffic patterns. Moreover, the time needed for the ANNs to produce the detections is measured. In order to succeed this, a delay model was added in the simulator. This delay model takes into consideration the time needed to finish the sampling, the time needed for the sample to reach the ANN and the time needed for the ANN to make the detection.

Time needed to finish the sampling is calculated based on the time the fault was injected and the time needed for the sampling completion. For example if the fault was injected at time 20 cycles, then 60 cycles are needed for the sampling completion (if the sampling period is 80 cycles). Time needed to reach the ANN is the number of hops needed for the sample to reach the ANN. For the time needed by the ANN to make the detection each neuron requires three cycles (one for the multiply operation, one for the accumulation and one for the LTU activation function).

Figure 5.10, summarizes the comparison when targeting the case study scenario of an 8x8 mesh NoC. For all the traffic patterns, the number of undetected / unexpected faults, in comparison to the correct detections, is very low (less than 4). Neighbor, Tornado and realistic traffic patterns show better results concerning the percentage of correct detections (98-99% and 97-99% respectively) and less delay (55 and 58 cycles respectively) compared to Transpose and Random traffic patterns (96-97%).

Added to this, simulations show that the miss detections for the cases of Neighbor, Tornado, Transpose traffic patterns are very low (less than 2). Random traffic pattern presents more miss detections but this is acceptable compared with the high percentage of correct detections.

5.3 ANN Costs and Power Consumption

For a 64-input ANN, a hyperbolic tangent activation function, and a single threshold sub-tractor (i.e. to perform a 64-input complete neuron operation to a single output from the activation function) the hardware costs are as follows:

Using Verilog and synthesized Synopsys Design Vision, targeting a 65 nm commercial CMOS library, at targeted frequency 500 MHz, at 1 V power supply voltage, using 20-input neurons, synthesis results indicate an estimated amount of 10,000 gates for a 20-input neuron (performing 5 parallel multiplications and accumulations per cycle).

Assuming 50% switching activity probability, the synthesized ANN described above consumes an estimated 0.00275 mW when computing one cycle of 20 inputs (one full router utilization packet). A total of 20 cycles is needed for multiply-accumulate, 1 cycle for thresholding, and 1 cycle for activation function lookup. Assuming that the ANN can start receiving data in Cycle 1, and with a steady flow of one utilization packet from each router (with 5 values enclosed) then the inputs needs 23 cycles in order to reach the next layer of neurons. So, 23 cycles * 0.00275 mW = 0.0633 mW in total, in order to reach the next layer of neurons. The next stage will of course use less power, but the same hardware can be re-used so there will be no need to compensate for extra area.

5.4 Comparison with Related works

Lastly, we briefly give a comparison with relevant related works for fault detection in NoCs in Table 5.3.

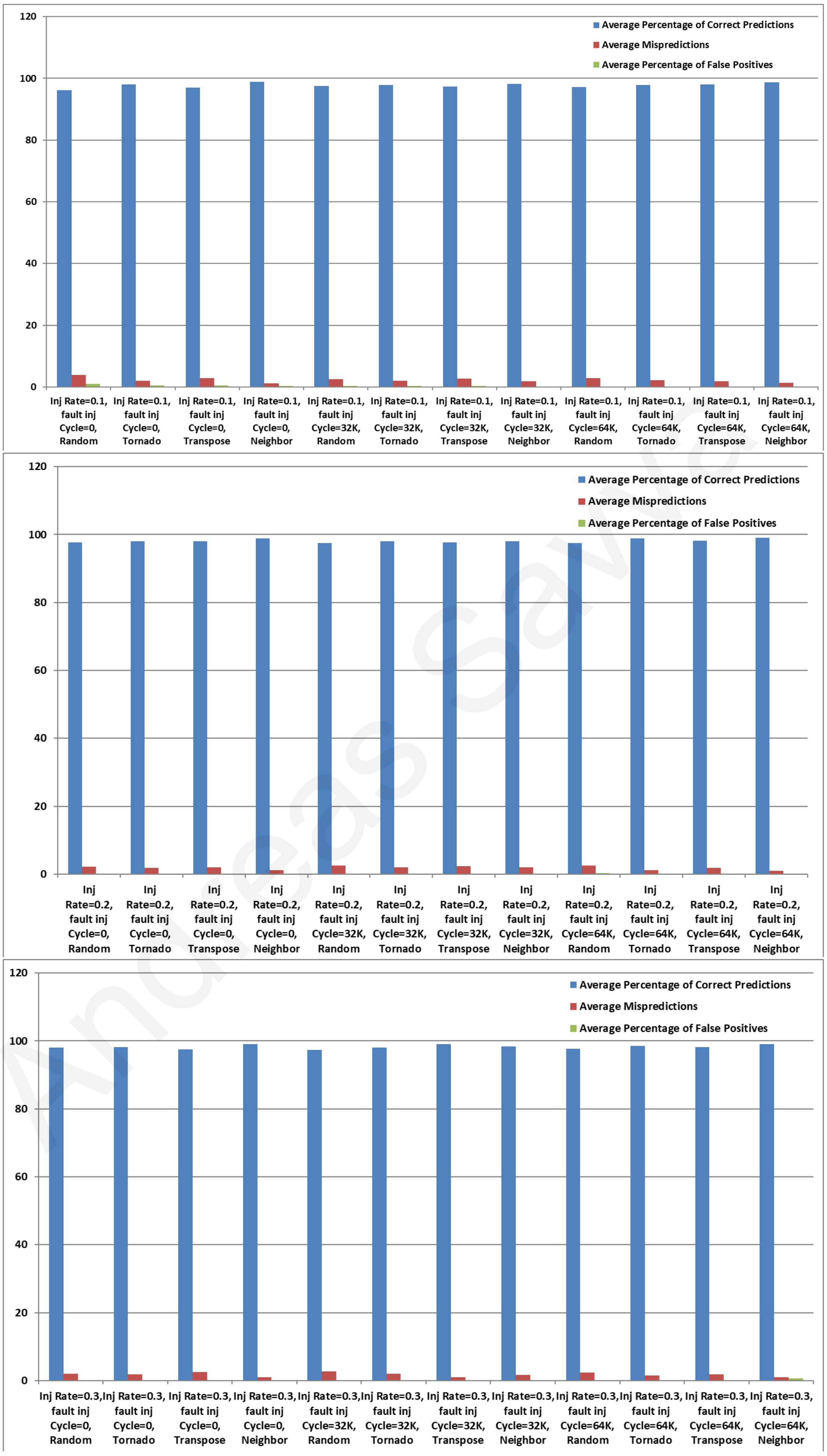


Figure 5.10: Detection results for different traffic patterns with three different injection rates (0.1 – 0.3) and three different fault injection cycles (0, 32K, 64K).

Table 5.3: Comparisons with related research.

Related Work	**Characteristics**	**Fault Detection**	**Hardware Overhead**
[7] STRN (Soft error Tolerant NoC Router) architecture	8x8 2-D mesh topology, Uniform and Tornado traffic	Detects most of the single soft errors	Additional area overheads (7%) and additional hardware overheads for the new NoC router architecture
[8] New fault detection architecture with Channel Tester and Detectors	Mesh topology, Modified framework and NoC routers	High level of fault tolerance for all the NoC links	Additional area overheads (5%) and additional hardware overheads for the modified NoC routers - Channel Tester
Proposed ANN-based Technique	8x8 2-D mesh topology, different traffic patterns	Up to 99% prediction accuracy for fault detection	10000 gates for a 20-input neuron (4% of the NoC hardware)

In [7], authors present Soft Error Tolerant NoC router (STRN) architecture. This architecture can detect soft errors in different control stages of the routing. This method can detect most of the soft errors with additional area overheads (around 7%) and additional hardware overheads for the additional NoC router architecture logic. Authors in [8], propose a new fault detection mechanism for NoC interconnects based on monitoring module – new fault detection architecture with channel testers and detectors. This work has high level fault tolerance for the NoC links with additional area and hardware overheads based on the modifications of the architecture and the channel testers and detectors. When compared to both [7] and [8], the ANN-based fault detection technique yields high level of fault detections as well (around 99% prediction accuracy), while still maintaining lower hardware overheads (less than 4% of the NoC hardware) when compared to both of them.

Through our intelligent method, we introduced the use of ANNs for fault prediction purposes. Based on their characteristics and their training process, ANNs helped us to achieve high level of fault prediction accuracy and keeping the additional hardware overheads minimum compared with other relevant works.

Additionally with the new proposed method, our work focuses on the analysis of the different aspects of the network in order to achieve optimal results. Our work focuses on the topological analysis of the ANN network and on the different parameters needed for the design of the framework for high level fault detection in networks. Our method can also be used for fault detection in different types of networks and hardware based on the scalable nature of the calculations. By using real-time data for training our method offers better insights and more accurate results making our work more competitive against other works.

5.5 Conclusion

This chapter presented a design exploration framework for fault detection in hardware systems with the use of high level ANNs. It analyzed, explained and evaluated all the necessary steps taken in designing such a mechanism. In order to evaluate the ANNs, a NoC case study is used. Based on their ability to dynamically be trained ANNs can be used for the detection of inter-router link faults in NoCs. Based on the experiments, two important things need to be analyzed and studied, the ANN topology/design and the network. For the ANN topology/design, a lot of experiments are developed in order to choose the appropriate ANN architecture and internal ANN parameters such as neurons in the input/hidden/output layers. From the above analyzed experiments, based on simulated results, 4x5 NoCs/ANNs with 19 neurons in hidden layer were chosen for this work. Based on experiments on 8x8 NoC networks, different simulator parameters were explored and the average router ports utilization values were developed for the ANN training phase. Added to this, an efficient delay model was implemented in the simulator for comparison purposes.

The ANN utilizes very low hardware resources and can be integrated in larger hardware systems easily. Simulation results show good detection results up to 99% under synthetic and realistic traffic models. Thus, it can be concluded that by designing correctly the ANNs can be very beneficial for detecting faults in networks, especially in large and complex systems such as NoCs.

Chapter 6

Robustness of ANNs Based on Weight Alterations Used for Prediction Purposes

6.1 Introduction

Nowadays, due to their excellent prediction capabilities, the use of artificial neural networks (ANNs) in software has significantly increased. One of the most important aspects of ANNs is robustness. Most existing studies on robustness focus on adversarial attacks and complete redundancy schemes in ANNs. Such redundancy methods for robustness are not easily applicable in modern embedded systems. This work presents a study, based on simulations, about the robustness of ANNs used for prediction purposes based on weight alterations. We devise a method to increase the robustness of ANNs directly from ANN characteristics. By using this method, only the most important neurons and connections are replicated, keeping the additional hardware overheads to a minimum. For implementation and evaluation purposes, the networks-on-chip (NoC) case, which is the next generation of system-on-chip, was used as a case study. The proposed study/method was validated using simulations and can be used for larger and different types of networks and hardware due to its scalable nature. The simulation results obtained using different PARSEC (Princeton Application Repository for Shared-Memory Computers) benchmark suite traffic show that a high level of robustness can be achieved with minimum hardware requirements in comparison to other works.

6.2 ANNs and Robustness

Neural networks yield excellent prediction results if appropriately trained for use in different application domains [64]. They have the power to extract valuable information from complex data and predict trends that cannot be easily detected by other mechanisms. While neural networks have certain prediction capabilities, their accuracy decreases in the presence of small perturbations. This makes it difficult to apply them in critical areas [10].

One of the main research goals of the analysis of robustness is to propose different solutions/architectures with increased robustness [11]. This is one of the fundamental problems that require extensive future research since ANN faults significantly affect the accuracy and reliability of these types of networks.

One of the most well-known solutions for robustness improvement and fault tolerance is to apply triple/dual modular redundancy (n-MR schemes) [48], replicating the entire ANN architecture, but these methods are very restrictive due to additional hardware overheads. This redundancy includes full replications of ANN architecture and is not applicable to modern on-chip systems due to area limitations [66].

Additionally, the robustness of neural networks to adversarial attacks is critical due to security issues that make these neural networks vulnerable [46], [47], thus causing poor performance and accuracy. Until recently, researchers concentrated on comparing results based on having/not having adversarial machine learning attacks and providing different solutions. Defenses based on adversarial training have been proposed, but these defenses are often defeated by stronger attacks [47].

Motivated by the above, we focused on the property of robustness of neural networks used for prediction purposes based on weight alterations. Therefore, we examined how the prediction accuracy of ANNs is impacted by weight faults. The networks-on-chip case, which was presented in the framework chapter, was used as a case study to evaluate the robustness of the ANNs based on simulations. The goal of this work is to discuss the robustness of neural networks to changes in weights that might affect the prediction results. This work discusses and analyzes weight alterations in ANNs based on simulations/implementations and provides a protection/robustness method for ANNs. This method will help to maintain high robustness in ANNs with minimum additional hardware overheads. In this work, the architecture of the ANNs was changed by duplicating only the most important neurons/connections in order to achieve good prediction accuracy for ANNs weight faults. We analyzed the importance of neurons/connections in ANNs based on actual simulations/implementations and how the prediction accuracy of the ANNs is affected in cases of weight faults.

The rest of this chapter is organized as follows. Section 3 introduces the methodology and a robust approach for detecting weight faults in ANNs with the support of simulation results and analysis. Section 4 offers a brief conclusion to the chapter.

6.3 Methodoogy

6.3.1 Development of the ANNs and Network Traffic

In order to verify the robustness of the developed ANNs, we studied how prediction accuracy is affected by weight alterations in ANNs. For the implementation of the ANNs, we used MATLAB and the nntool. PARSEC benchmark suite was used for the traffic requirements [60]. To collect the ANN training data for predictions, we used NoC as a case study. NoC is an emerging technology that provides high-bandwidth and low-power on-chip communication between many cores. For the purposes of this work and for scalability purposes, we partitioned the NoC topology into smaller parts, and four different ANNs were created, one for each individual partition of NoC topology based on the explanations provided in our previous research/chapters [64]. The monitoring of the entire NoC topology/partitions is carried out in parallel using each developed ANN. Each of the four developed ANNs is responsible for monitoring one NoC partition, and the ANN receives data from the specific NoC partition that is used for the training process. The ANNs can be considered independent processing units in NoC topology. Different-sized ANNs were studied, and these have the same structure/architecture depending on the size of the NoC; they differ only in the weights. Figure 6.1 presents the structure of the neural network.

In this work, integrated hardware-based ANNs with 19 neurons in the hidden layer were developed and, based on the ANN training and data received from the NoC simulator, intelligently predicted which routers might present fault, as explained in [64]. We used ANNs with one hidden layer, which is sufficient since

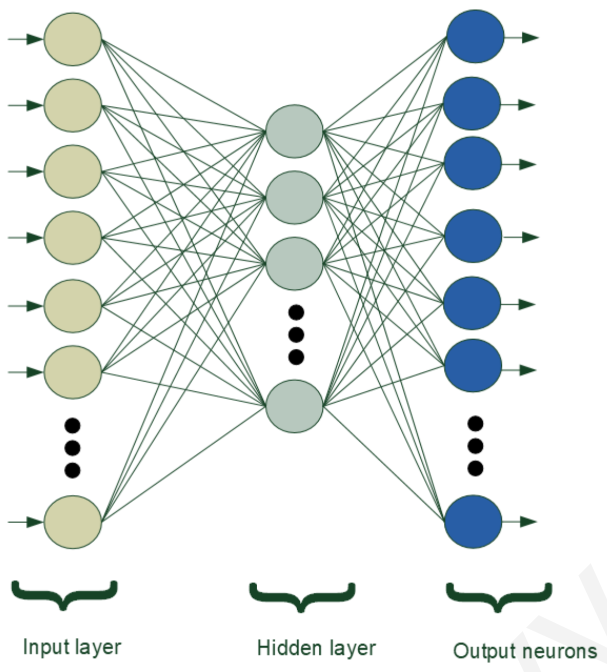


Figure 6.1: Structure of the neural network.

more hidden layers can introduce a risk of converging to a local minimum and might not improve the model [64]. For the training stage of the developed feed-forward ANNs, we used a back-propagation training algorithm, and the training was carried out offline.

For this work, the size of the ANNs is small, and based on the scalable nature of the calculations, the results are indicative for larger NoCs/ANNs. By using a simple activation function and having a small number of neurons for each ANN, we managed to keep the implementation of the ANNs simple. Each feed-forward ANN uses a hyperbolic tangent as an activation function, which provides simplicity and accuracy. A hyperbolic tangent produces outputs at a scale of [-1, 1]; it is a continuous, symmetric, and asymptotic function that is responsible for processing the output of each neuron in order to feed it to the next adjacent layer.

In the next sections, we describe the verification of how the prediction accuracy of ANNs was affected by weight alterations using simulations. Based on the results, we propose a method to maintain robustness in accepted margins with minimum additional overheads.

Additionally, network performance and a thorough analysis of the different implementations highly depend on network traffic. In order to achieve more accurate results for our simulations, we used data from the PARSEC benchmark suite, which provides realistic traffic profiles, traces of real applications based on parallel programs, and state-of-the-art algorithms that help with the study of the implemented topologies [60], [63], [67].

6.3.2 Simulations to Verify How the Prediction Accuracy of the ANNs is Generally Affected

Firstly, different random simulations were developed to verify whether weight alterations affect the percentage of correct predictions of the ANNs in general. One NoC topology/partition was randomly chosen, a dedicated ANN is responsible for this partition, and random ANN weight alterations were injected and simulated. Different simulation cases were developed using one, two, three, and four different weight alterations in the case of 8×8 NoC topologies with 4×5 NoC partitions/ANN sizes. For each weight, one random bit was chosen to be altered. All the different NoC partitions and ANNs were verified in this work. As a starting point, we only used 8×8 NoC topologies with 4×5 partitions/ANN sizes and changed one bit for each weight in order to verify if the percentage of correct predictions is generally affected. In the following sections, we present more simulations of alterations for different ANN sizes.

Figures 6.2 and 6.3 present the results for one, two, three, and four weight alterations in the developed ANNs. Figure 6.2 presents the results from the simulations in the case of checking all the weights of the ANN: one bit alteration. Figure 6.3 shows the results in the case of checking the input weights only: one bit alteration. The results show that there is a clear decrease in percentages of the correct predictions of the different simulated cases.

Based on the graphs/results, we conclude that weight alterations impact the prediction accuracy of the ANNs. Since the accuracy whereby only the input weight alterations are checked decreased below 90% on average, and the accuracy whereby all the weight alterations are checked is above 90% on average, we can conclude that if the alteration is presented in the input weights, the impact on accuracy is larger than if the alteration is presented in the remaining weights. This allows us to assume that the input-hidden connections of the ANN are more influential in the prediction process than the hidden-output connections of the ANN.

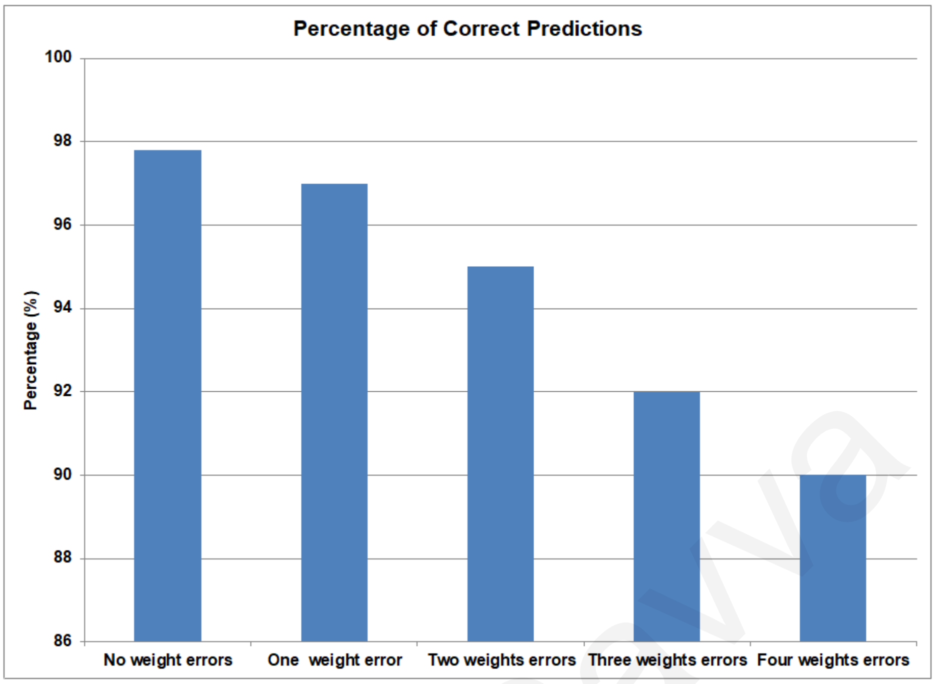


Figure 6.2: General results for checking all weights: one bit error.

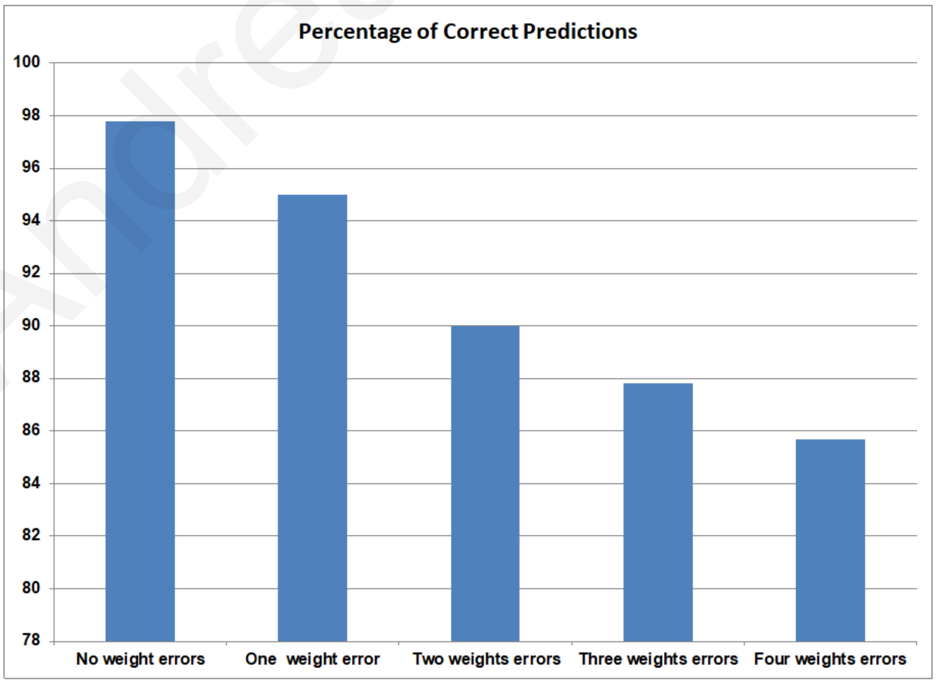


Figure 6.3: General results for checking only input weights: one bit error.

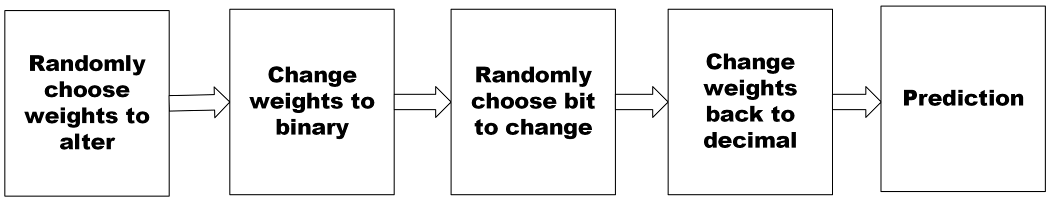

Randomly choose weights to alter
Change weights to binary
Randomly choose bit to change
Change weights back to decimal
Prediction

Figure 6.4: Flowchart presenting the process of weight alterations.

6.3.3 More In-depth Simulations and Explanations for the Robustness of the ANNs Based on Weight Alterations

Next, we checked how the alteration of weights impacts the overall prediction process for each of the different implemented ANNs in more detail. In order to achieve this, different sizes of ANN architectures were developed, and random weight alterations were injected into all of the individual developed ANNs. All the individual ANNs for each partition of the different NoC topologies and all the weights were checked. Randomly chosen bit alteration of the randomly chosen weight/weights were checked in this work. Figure 6.4 presents the weight alteration process, and explanations are provided below.

An exhaustive testing process of all the random possibilities of the weight faults is developed for each individual ANN. Thousands of different simulations are developed based on randomly chosen weights and bit for all ANNs. To generate erroneous weights, we tested all the possible random-weight bit alterations for one or more weight errors. For each of these simulation cases, we used PARSEC workloads. All the results were recorded, and based on the prediction results, we managed to conclude on the most important neurons/connections of the ANNs. For the purposes of our method, a neuron/connection is defined as important if the weight/bit alterations cause the ANN to yield erroneous prediction results. Next, we present some of the different simulation cases (Table 6.1) with results.

Different sizes of the NoCs/ANNs were developed and simulated. We started working with one randomly chosen network-on-chip partition: a dedicated ANN is responsible for each individual partition. All network partitions and different workloads from the PARSEC benchmark suite were verified and presented. Different simulation cases were developed and checked. Table 6.1 presents some of the developed simulation cases with appropriate descriptions. Figure 6.5 shows the results concerning the percentage of correct predictions for each of the above-

Table 6.1: Explanations for different simulation cases.

Simulation Case	**Description**
Case A: One randomly chosen weight error and one randomly chosen bit alteration.	Firstly, we randomly choose one weight. The weight is changed to binary. After this, we randomly chose one bit to be altered. The bit is changed. We change the altered weight back to decimals and continue with the predictions.
Case B: Two randomly chosen weight errors and one randomly chosen bit alteration.	Firstly, we randomly choose two weights. The weights are changed to binary. After this, we randomly chose one bit to be altered for each weight. The bits are changed. We change the altered weights back to decimals and continue with the predictions.
Case C: Three randomly chosen weights and one randomly chosen bit alteration.	Firstly, we randomly choose three weights. The weights are changed to binary. After this, we randomly chose one bit to be altered for each weight. The bits are changed. We change the altered weights back to decimals and continue with the predictions.
Case D: Four randomly chosen weights and one randomly chosen bit alteration.	Firstly, we randomly choose four weights. The weights are changed to binary. After this, we randomly chose one bit to be altered for each weight. The bits are changed. We change the altered weights back to decimals and continue with the predictions.

mentioned cases in comparison with the case of no weight faults for different ANN sizes. From the results, we can conclude that weight changes in the ANNs impact the whole prediction process, and we used thousands of additional simulations to determine the most important ANN neurons for each case (partition and PARSEC workload). In the following section, we present and analyze a method/technique for maintaining the robustness of ANNs in high levels based on the replication of the most important neurons/connections.

The amount of redundancy that is needed to achieve good levels of robustness is usually high in overheads. If less redundancy is added, this means that fewer faults will be tolerated. A balance between this redundancy and robustness in additional overheads is needed. Based on our research, we attempted to maintain the robustness of ANNs in accepted margins and have as few overheads as possible by only duplicating the most important neurons/connections. Our method is presented in the next section.

6.3.4 Replication of the Most Important Neurons/Connections

Next, we present a method in order to achieve robustness for ANNs based on the replication of the most important neurons/connections, addressing the issue of robustness based on redundancy.

Different methods have previously been used for robustness, but these are often restrictive. Most cases are based on replicating complete ANN architecture or replicating a certain number of complete layers in the ANNs. Based on past studies, ANNs are not always fault-tolerant and indicate the need for more robust methods [10], [68].

Based on the results presented in the previous section for each individual ANN, as well as the PARSEC benchmark suite workloads, we managed to identify the most important neurons/connections in the presence of weight (bit) errors. Our method focuses on the replication of these parts in order to achieve robustness for the developed ANNs.

In order to check how many redundant neurons/connections are needed for the different ANN sizes to achieve good robustness levels, different numbers of neurons were replicated and simulated. Figure 6.6 presents the results from simulations of different numbers of neuron replications for different ANN sizes.

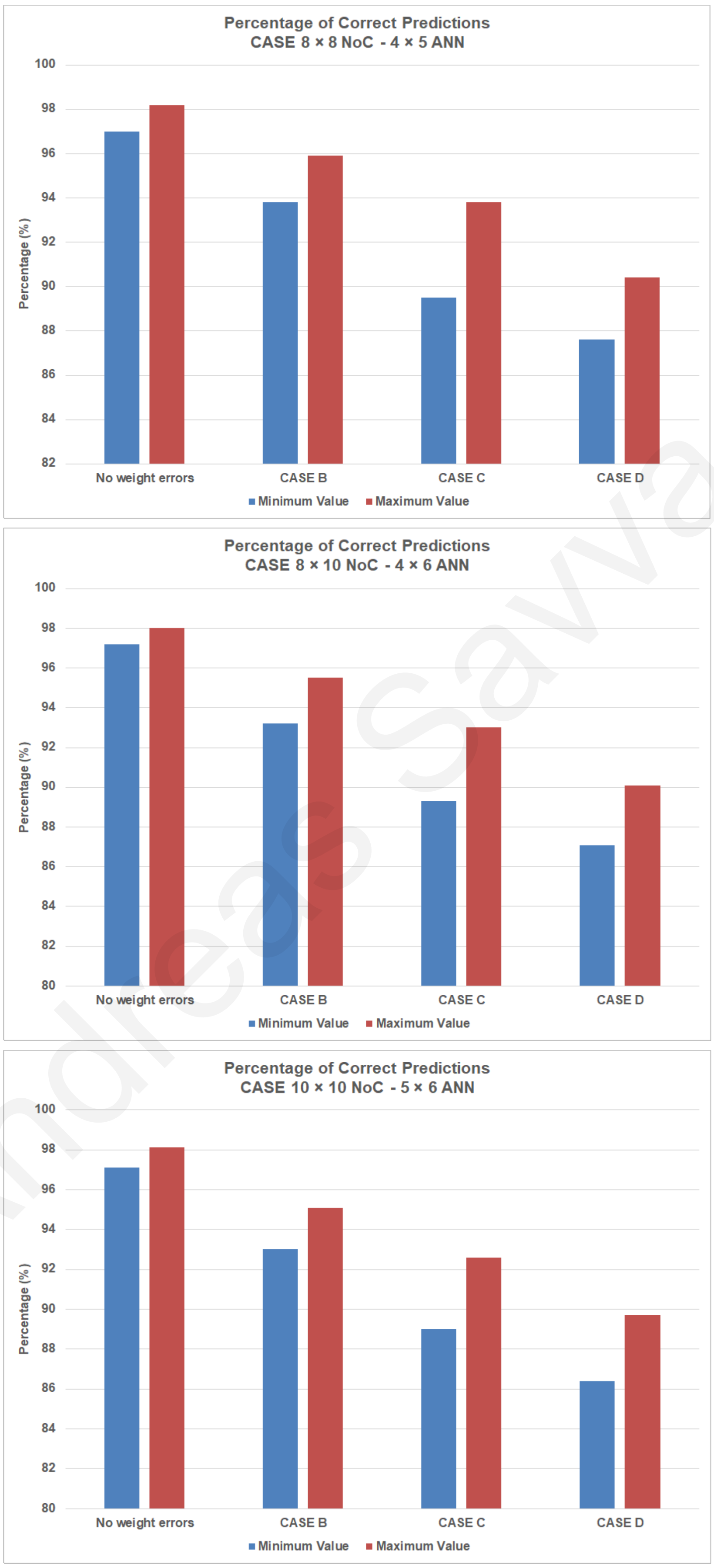


Figure 6.5: Percentage of correct predictions for different weight/bit alteration cases and different ANN sizes.

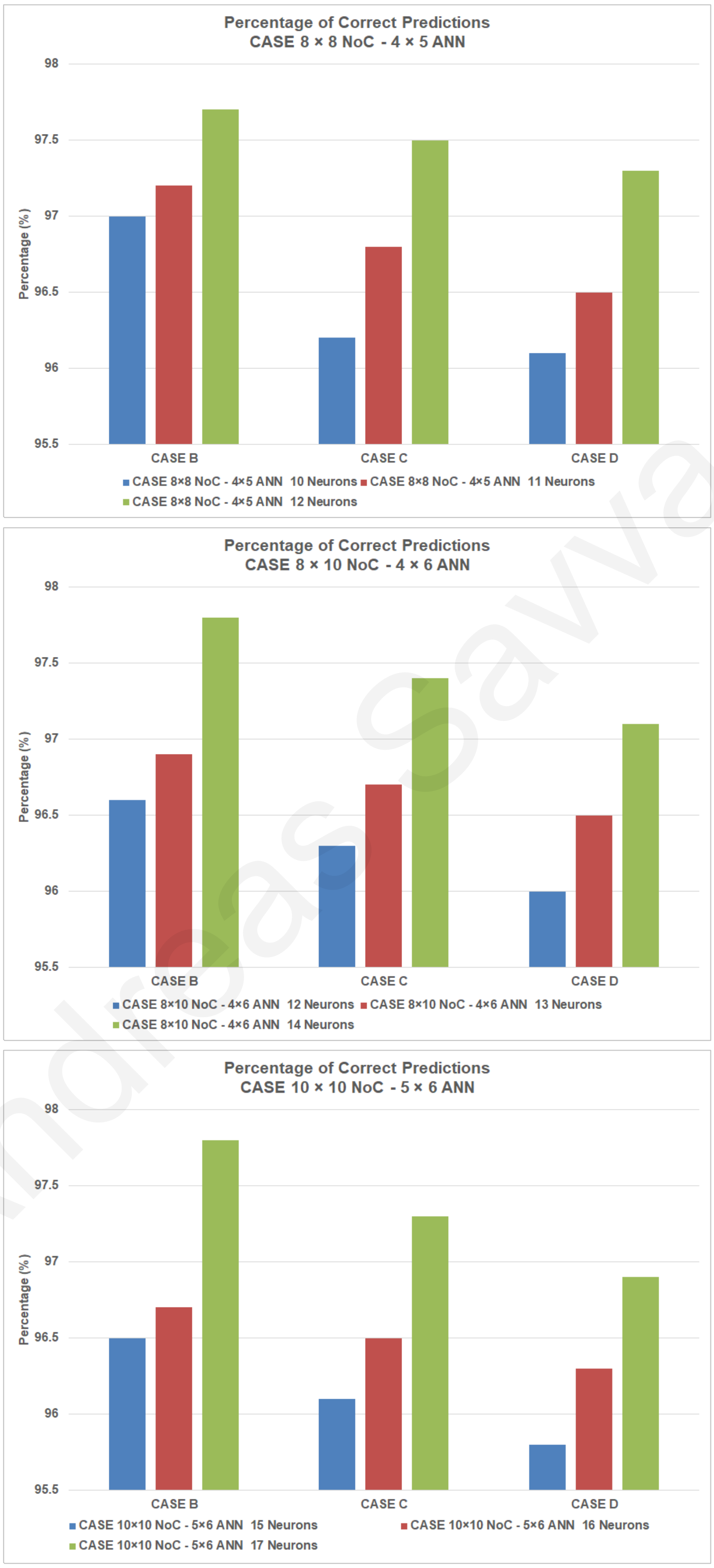


Figure 6.6: Simulation results for different ANN sizes and different numbers of neuron replication.

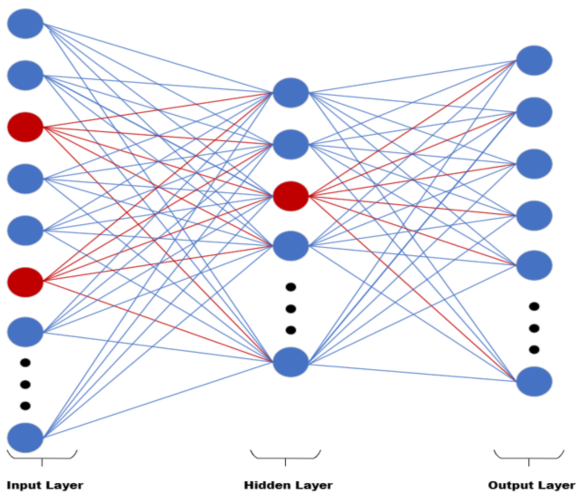


Figure 6.7: Structure of the neural network with redundant neurons/connections.

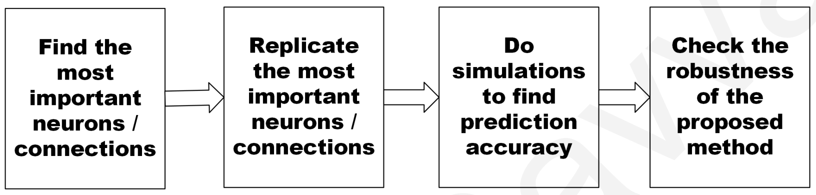

Find the most important neurons / connections
Replicate the most important neurons / connections
Do simulations to find prediction accuracy
Check the robustness of the proposed method

Figure 6.8: Steps for replicating the most important neurons/connections for robustness.

The results show that good robustness results are achieved for the 4 × 5 ANN with 12 neuron replications. For the 4 × 6 ANN, good results are achieved with 14 neuron replications. For the 5 × 6 ANN, good robustness results are achieved with 17 neuron replications. Figure 6.7 presents the structure of the neural network with redundant neurons/connections (redundant neurons/connections are shown in red).

Figure 6.8 shows the complete process/steps of our method.

Based on the previous results, since the weight alterations are already known, we determined the most important neurons/connections for specific ANNs under different PARSEC workloads. Different simulations are developed with randomly chosen weights and bit, as explained before. We tested all the possible random weight bit alterations for one and more weight errors. Next, we replicated the above important neurons/connections of the specific ANN, changing the architecture of the ANN in order to increase its robustness. Based on the new and altered ANN architecture, we carried out simulations to check the prediction accuracy of different ANNs. Using the new prediction simulation results, we verified the robustness of the ANNs.

Figure 6.9 shows that the robust method analyzed above (with redundant neu-

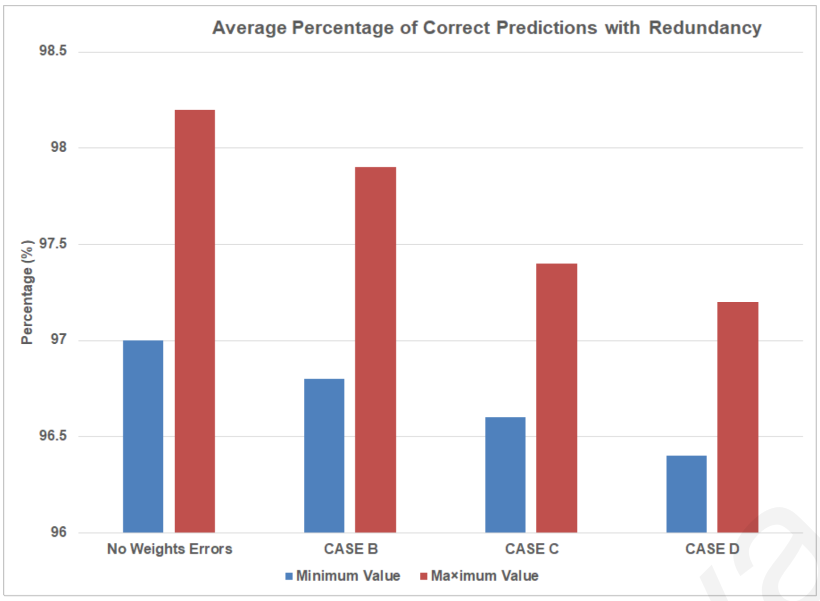


Figure 6.9: Average percentage of correct predictions for different weight/bit alteration cases with redundant neurons/connections.

rons/connections) helps to increase the robustness of the ANNs used for prediction purposes. The figure shows the average percentages of correct predictions for each of the above-mentioned cases with redundancy in comparison to cases with no weight faults.

In our method, robustness is achieved via the actual ANN process itself by using the high connectivity of the neurons in the ANNs. The above results show that, by replicating the most important neurons/connections, ANNs can retain a high robustness level (around 98% prediction accuracy); these results are reproducible for larger networks too.

6.4 Comparison with Related Works

Lastly, we provide a brief comparison with relevant related studies, as shown in Table 6.2.

Authors in [48], propose a fine-grained redundancy framework D2NN based on dual modular redundancy for hardware vulnerabilities. The DMR consists from the primary network – the original one and a secondary network is constructed using newly introduced neurons together with parts of the original DNN. Based on this framework authors shown good robustness levels but the hardware overheads are high because of the duplicated network. Authors in [51] present Embedded

Table 6.2: Comparisons with related research.

Related Work	**Overheads**	**Robustness Level**
D2NN [48] (Dual modular redundancy framework)	DMR—Primary and secondary networks—The primary DNN represents the original DNN, while the secondary DNN is constructed using newly introduced neurons together with parts of the original DNN.	Faults on the secondary network are converted from missed to detected faults, and the fault miss rate is reduced, maintaining good robust levels.
E2CNNs [51] (Convolutional neural networks: ensemble architectures)	The group of architectures is built by training pruned CNN several times. The individual predictions are averaged together to compute the output. (Additional overheads are needed for the control logic).	A 4-E2CNN increases the accuracy around 15%.
Our work (NoCs—different cases of PARSEC benchmark suite)	Only the most important neurons/connections are replicated, keeping the overheads to a minimum.	Based on simulated results, high levels of robustness are achieved: around 98% prediction accuracy.

Ensemble CNNs (Convolutional neural networks) – E2CNNs. The ensemble is built by training a pruned CNN several times. Then the individual predictions are averaged together to compute the output. This method presents additional overheads for the control logic and an increase in the accuracy (around 15%). When compared to both [48], [51], our work yields high levels of robustness (around 98% prediction accuracy) while still maintaining lower hardware overheads.

The number of input–hidden–output units remains the same in our study, with only the most important neurons/connections changing. By replicating only these, we kept the overheads to a minimum in comparison to the previous studies that replicated the complete ANN or complete layers x times. Furthermore, our study not only considered single-fault assumptions (as in previous research) but verified all cases of random faults with one and more weights and single bit errors.

Moreover, our method tries to reduce the redundancy needed (in comparison to previous methods) in order to minimize expenses in terms of the additional units and links. Our method is easier to implement in practice in comparison to the other methods that require replications of more neurons / connections (even entire ANNs), and this makes them impractical in reality.

6.5 Conclusions

This chapter offers fundamental insights into the robustness of ANNs that are subjected to weight alterations and used for prediction purposes. For simulation purposes, NoC was used as the case study. Thousands of different simulations were created for different NoC/ANN sizes and weight/bit alterations in order to verify how the prediction accuracy of the ANNs is affected. In this work, we verified all the cases of random faults with one or more weight and one bit errors. Based on this study, we managed to draw significant conclusions on the robustness of the ANNs based on weight alterations. Additionally, we proposed a robustness method for ANNs based on redundancy. More specifically, the most important neurons/connections, which were defined based on simulations, were replicated. The proposed method can be used for larger networks and different hardware due to its scalable nature, and robustness is achieved through the actual ANN process itself. Our results indicate that a significant amount of redundancy is needed in order to achieve good levels of robustness in ANNs. Based on the analytical results from our simulations, we conclude that the proposed method can maintain the robustness of ANNs at high levels (around 98% prediction accuracy). Lastly, our method minimizes additional redundant units and links, keeping the additional hardware overheads and costs to a minimum.

Chapter 7

Conclusions and Future Directions

This chapter concludes all topics discussed in this dissertation, along with the key outcomes achieved by the techniques proposed in the thesis. Moreover, interesting future research directions regarding the topics considered in this dissertation are also presented.

7.1 Conclusions

The objective of this dissertation is to design and implement intelligent monitoring techniques for hardware using Artificial Neural Networks. As case study for simulation and evaluation purposes, the NoC case is used.

This work firstly presents a design exploration framework for new intelligent techniques with the use of high-level ANNs for hardware systems (NoCs) which will be used for evaluation purposes. It analyzes, explains and evaluates all the necessary steps taken for the development of an optimal framework. Lots of different parameters are analyzed and explained based on simulations. Based on experiments, two important things are analyzed and studied, the ANN topology/design and the NoC. For the ANN topology/design, a lot of experiments are developed in order to choose the appropriate ANN architecture and internal ANN parameters such as neurons in the input/hidden/output layers. From the above analyzed experiments, based on simulated results, 4x5 NoCs/ANNs with 19 neurons in hidden layer were chosen. Based on experiments on 8x8 NoC networks, different simulator parameters were explored and optimally analyzed and the average router link utilization values were developed for the ANN training phase. Added to this, an efficient delay model was implemented in the simulator for comparison purposes. The framework is used for the development of new intelligent mechanisms for hardware/NoCs.

Next, this work continued with the use of the framework for the implementation of an ANN-based intelligent power management policy. This dynamic method presents how an ANN based mechanism can be used to intelligently select, based on appropriate training, candidate links to be turned off and then back on in order to achieve power savings. The ANN based mechanism can identify a significant amount of future behavior of the system therefore it can intelligently select candidate links for turn off. The ANN monitors the link utilization values within its region. These values are processed by the ANN which computes the links that should be turned off during each interval – underutilized links are turned off. The ANN-based model utilizes very low hardware resources, and can be integrated in large mesh and torus NoCs and hardware, exhibiting significant power savings. Simulation results indicate approximately 13% additional power savings when compared to a statically-determined threshold methodology under synthetic and realistic traffic models at very low hardware overheads (less than 4%). Through this work, power savings are achieved through the use of small and scalable ANNs which gives an advantage against statical ways making our work flexible under any application is required to facilitate. Additionally, static power consumption is minimized by turning links off and then back on.

This work additionally presented an intelligent mechanism for fault detection in hardware systems - NoCs with the use of high level ANNs. The ANNs, based on dynamic training, can be used for detection of inter-router link faults. This work, analyzes, explains and evaluates all the necessary steps taken in designing such a mechanism. The new intelligent method can be used for fault detection in different types of hardware and networks based on the scalable nature of the calculations. The ANN utilizes very low hardware resources and can be integrated in larger hardware systems easily. Additionally, by using real time data for the training, this mechanism offers better insights and more accurate results making our work more competitive against other works. Simulation results under synthetic and realistic traffic models show good detection results up to 99% with a delay less than 60 cycles. Synthesis results indicate an estimated amount of 0.0633 mW power consumption per neuron for the implemented ANN when computing a complete cycle. Thus, it can be concluded that by designing correctly the ANNs can be very beneficial for intelligently detecting faults in hardware, especially in large and complex systems such as NoCs.

Lastly, this work developed an intelligent robust technique for the ANNs. Discussion/study about the robustness of neural networks to changes of weights that might affect the prediction results is provided and a new technique based on topology redundancy – architectural alteration of the ANN is proposed and evaluated based on simulations. Based on simulations results, we concluded that weights alterations impact the prediction accuracy of the ANNs. Through those results we managed also to conclude that the input weights of the ANN have more impact on the prediction accuracy than the rest of the weights. In-depth analysis and simulations about how the alteration of weight impact the overall prediction accuracy are presented and based on the results a new robust method is developed for the ANNs. The new method introduces redundancy of the most important neurons (and connections) only, and those are defined based on simulations. In this method, robustness is achieved through the actual ANN process itself. We take into consideration the advantages of high connectivity and the rest of the ANNs attributes which are important characteristics of the ANNs. Added to this, our method tries to minimize the redundancy needed in comparison with other methods in order to minimize the expenses on terms of the additional units and links needed. Our method is evaluated based on simulations and we concluded that the proposed method keeps the robustness of ANNs in high levels with minimum hardware overheads. Compared with other techniques, this method is easier to implement since it replicates only the most important neurons minimizing the overheads in comparison with previous works that replicate complete architectures or layers, making them impractical in reality.

7.2 Future Research Directions

The combination of power management and fault detection techniques in hardware is a critical area of research due to the increased complexity and area limitations in hardware.

Dynamic link power management techniques adjust the link characteristics based on the link utilization in order to achieve power savings. Fault detection techniques can be used along with these techniques to monitor the performance of the hardware and predict future faults. Additionally, underutilized links can be turned off and then back on in order to conserve power. As previously mentioned, fault detection mechanisms can continue monitoring the hardware in order to detect future faults, in parallel with the turning off of the links. Moreover, proactive fault detection methods have the ability to find errors based on past data or based on forecast analysis. These methods can enable the implementation of power management solutions as well. Based on this, hardware can predict and prevent faults and achieve good power savings.

Overall, power management and fault detection mechanisms for hardware allow the development of more efficient and reliable hardware with the ability to predict future faults and having good levels of power savings at the same time.

There are a number of intriguing future research directions that are interesting regarding the contents suggested through this dissertation. Future research could concentrate on the improvement of the different solutions mentioned through this work for the power consumption and fault detection of hardware and robustness of the ANNs and applying the specific ideas in different types of newer and more complex hardware. Additionally, the usage of the ANNs could be further explored for the purposes of fault predictions and power management in hardware. Based on this, future work can explore the usage of ANNs in order to predict the most power efficient routing paths and this can lead to new power-aware strategies for the minimization of the power consumption in hardware. The next sections provide a brief description of different future research directions.

7.2.1 CPU Prediction and Hardware Vulnerabilities Prediction

CPU (Central Processing Unit) is one of the most important sources of measuring the performance of a system and has the highest levels of demands. Our work can be used for the monitoring of the CPU and for predicting the future CPU utilization/performance and CPU failures with great accuracy based on appropriate training with the use of CPU utilization data. ANNs can be developed based on the guidelines provided in the previous chapters in order to achieve accurate predictions. The developed ANNs can be used to learn the relationship between CPU performance and already existed workloads or new ones (- real time) and for predicting CPU failures. The ANN mechanism can monitor the utilization values of the

CPU and these values can be used from the ANN in order to make the prediction of faults. The ANN can be trained based on utilization values collected from the CPU in which random faults are injected. Based on the training phase, the ANN can detect future CPU faults.

Moreover, our work can be used for the prediction of different hardware vulnerabilities and faults. ANNs can be used for the fault prediction by finding different patterns in the utilization data based on appropriate training as previously mentioned. With the use of this information, researchers can decide about hardware replacements of malfunctioning components in order to increase the performance and availability of the different hardware. The developed ANNs are adaptable in many different hardware systems.

Additionally, our work, based on appropriate training, can predict future hosts machines failures and disk utilizations/failures. Based on the ideas of our work, ANNs can be used to intelligently monitor the host machines and the disks utilizations and based on these values, the ANNs can proceed with the prediction of failures. Developed ANNs can be trained off-line based on values collected from devices in which random faults are injected. The ANNs then, based on the training, can intelligently detect faults in host machines and disks and researchers could conclude on decisions for these machines/devices – for example perform off/on host machines to deal with future demands or replacing the faulty host machines.

7.2.2 Memory Fault Prediction

Memory errors can lead to failures and the memory failure analysis is one of the most important aspects in hardware reliability. Our work can be used as a mechanism for predicting future memory failures. We can observe important aspects of memory failures, study the impact of memory failures and propose new mechanisms based on the guidelines provided in this dissertation. We can use the ANNs in order to find patterns and associations in the memory data for accurate fault predictions. The developed ANNs can be trained based on real-time memory data (– various memory parameters) in which random faults are injected. After the training, which can be done off-line when the ANN is not in use, the ANN can intelligently predict future memory faults. By repeatedly taking the memory failure data as training data, we can predict memory failures in the future. After the prediction, error prevention mechanisms such as dynamic substitution with spare memory components can be proposed.

Bibliography

[1] A. Jain, J. Mao, and K. Mohiuddin, “Artificial neural networks: A tutorial”, *Computer*, vol. 29, no. 3, pp. 31–44, 1996. DOI: `10.1109/2.485891`.

[2] R. Schalkoff, *Artificial Neural Networks* (McGraw-Hill series in computer science). McGraw-Hill, 1997, ISBN: 9780070428072. [Online]. Available: `https://books.google.com.cy/books?id=bbNQAAAAMAAJ`.

[3] R. A. Shafik, J. Mathew, and D. K. Pradhan, “Introduction to energy-efficient fault-tolerant systems”, in *Energy-Efficient Fault-Tolerant Systems*, J. Mathew, R. A. Shafik, and D. K. Pradhan, Eds. New York, NY: Springer New York, 2014, pp. 1–10, ISBN: 978-1-4614-4193-9. DOI: `10.1007/978-1-4614-4193-9_1`. [Online]. Available: `https://doi.org/10.1007/978-1-4614-4193-9_1`.

[4] B. Hoefflinger, “Itrs: The international technology roadmap for semiconductors”, in *Chips 2020: A Guide to the Future of Nanoelectronics*, B. Hoefflinger, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 161–174, ISBN: 978-3-642-23096-7. DOI: `10.1007/978-3-642-23096-7_7`. [Online]. Available: `https://doi.org/10.1007/978-3-642-23096-7_7`.

[5] S. Murali, T. Theocharides, N. Vijaykrishnan, M. Irwin, L. Benini, and G. De Micheli, “Analysis of error recovery schemes for networks on chips”, *IEEE Design Test of Computers*, vol. 22, no. 5, pp. 434–442, 2005. DOI: `10.1109/MDT.2005.104`.

[6] C. Grecu, A. Ivanov, R. Saleh, E. Sogomonyan, and P. P. Pande, “On-line fault detection and location for noc interconnects”, in *12th IEEE International On-Line Testing Symposium (IOLTS'06)*, 2006, 6 pp.–. DOI: `10.1109/IOLTS.2006.44`.

[7] P. Poluri and A. Louri, “A soft error tolerant network-on-chip router pipeline for multi-core systems”, *IEEE Computer Architecture Letters*, vol. 14, no. 2, pp. 107–110, 2015. DOI: `10.1109/LCA.2014.2360686`.

[8] J. Liu, J. Harkin, Y. Li, and L. Maguire, “Online fault detection for networks-on-chip interconnect”, in *2014 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, 2014, pp. 31–38. DOI: `10.1109/AHS.2014.6880155`.

[9] C. Grecu, P. Pande, A. Ivanov, and R. Saleh, “Bist for network-on-chip interconnect infrastructures”, in *24th IEEE VLSI Test Symposium*, 2006, 6 pp.–35. DOI: `10.1109/VTS.2006.22`.

[10] N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks”, in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 39–57. DOI: `10.1109/SP.2017.49`.

[11] A. Fawzi, S.-M. Moosavi-Dezfooli, and P. Frossard, “The robustness of deep networks: A geometrical perspective”, *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 50–62, 2017. DOI: `10.1109/MSP.2017.2740965`.

[12] A. Savva, *Noc code*, Accessed: 8 April, 2024. [Online]. Available: https://www.github.com/Andsa1/NoC_Code.

[13] A. Savva, *Ann code*, Accessed: 8 April, 2024. [Online]. Available: https://www.github.com/Andsa1/ANN_Code.

[14] M. J. Islam, W. Senadeera, P. Brooks, R. Brown, R. Situ, P. Pham, and A. Masri, "An artificial neutral network (ann) model for predicting biodiesel kinetic viscosity as a function of temperature and chemical compositions", in *MODSIM2013, 20th International Congress on Modelling and Simulation*, Modelling, Simulation Society of Australia, and New Zealand Inc.(MSSANZ), 2013, pp. 1561–1567.

[15] V. Pacelli, V. Bevilacqua, M. Azzollini, *et al.*, "An artificial neural network model to forecast exchange rates", *Journal of Intelligent Learning Systems and Applications*, vol. 3, no. 02, p. 57, 2011.

[16] K. Suzuki, *Artificial neural networks: methodological advances and biomedical applications*. BoD–Books on Demand, 2011.

[17] Z. H. Khan, T. S. Alin, M. A. Hussain, *et al.*, "Price prediction of share market using artificial neural network (ann)", *International Journal of Computer Applications*, vol. 22, no. 2, pp. 42–47, 2011.

[18] S. Carrillo, J. Harkin, L. McDaid, S. Pande, S. Cawley, B. McGinley, and F. Morgan, "Advancing interconnect density for spiking neural network hardware implementations using traffic-aware adaptive network-on-chip routers", *Neural networks*, vol. 33, pp. 42–57, 2012.

[19] E. Alkım, E. Gürbüz, and E. Kılıç, "A fast and adaptive automated disease diagnosis method with an innovative neural network model", *Neural Networks*, vol. 33, pp. 88–96, 2012.

[20] R. C. Minnett, A. T. Smith, W. C. Lennon Jr, and R. Hecht-Nielsen, "Neural network tomography: Network replication from output surface geometry", *Neural Networks*, vol. 24, no. 5, pp. 484–492, 2011.

[21] M. H. Al Shamisi, A. H. Assi, and H. A. Hejase, "Using matlab to develop artificial neural network models for predicting global solar radiation in al ain city–uae", in *Engineering education and research using MATLAB*, Citeseer, 2011.

[22] A. G. Savva, T. Theocharides, and V. Soteriou, "Intelligent on/off link management for on-chip networks", in *2011 IEEE Computer Society Annual Symposium on VLSI*, IEEE, 2011, pp. 343–344.

[23] R. Marculescu, U. Y. Ogras, L.-S. Peh, N. E. Jerger, and Y. Hoskote, "Outstanding research problems in noc design: System, microarchitecture, and circuit perspectives", *IEEE Transactions on computer-aided design of integrated circuits and systems*, vol. 28, no. 1, pp. 3–21, 2008.

[24] G. Chen, F. Li, M. Kandemir, and M. J. Irwin, "Reducing noc energy consumption through compiler-directed channel voltage scaling", *ACM SIGPLAN Notices*, vol. 41, no. 6, pp. 193–203, 2006.

[25] T. Pering, T. Burd, and R. Brodersen, "Voltage scheduling in the iparm microprocessor system", in *Proceedings of the 2000 international symposium on Low power electronics and design*, 2000, pp. 96–101.

[26] F. Li, G. Chen, and M. Kandemir, "Compiler-directed voltage scaling on communication links for reducing power consumption", in *ICCAD-2005. IEEE/ACM International Conference on Computer-Aided Design, 2005.*, IEEE, 2005, pp. 456–460.

[27] V. Soteriou, N. Eisley, and L.-S. Peh, "Software-directed power-aware interconnection networks", *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 4, no. 1, 5–es, 2007.

[28] E.-Y. Chung, G. De Micheli, and L. Benini, "Contents provider-assisted dynamic voltage scaling for low energy multimedia applications", in *Proceedings of the 2002 international symposium on Low power electronics and design*, 2002, pp. 42–47.

[29] J. Kim and M. A. Horowitz, "Adaptive supply serial links with sub-1-v operation and per-pin clock recovery", *IEEE Journal of Solid-State Circuits*, vol. 37, no. 11, pp. 1403–1413, 2002.

[30] L. Shang, L.-S. Peh, and N. K. Jha, "Dynamic voltage scaling with links for power optimization of interconnection networks", in *The Ninth International Symposium on High-Performance Computer Architecture, 2003. HPCA-9 2003. Proceedings.*, IEEE, 2003, pp. 91–102.

[31] D. Shin and J. Kim, "Power-aware communication optimization for networks-on-chips with voltage scalable links", in *Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, 2004, pp. 170–175.

[32] V. Soteriou and L.-S. Peh, "Dynamic power management for power optimization of interconnection networks using on/off links", in *11th Symposium on High Performance Interconnects, 2003. Proceedings.*, IEEE, 2003, pp. 15–20.

[33] V. Soteriou and L.-S. Peh, "Exploring the design space of self-regulating power-aware on/off interconnection networks", *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 3, pp. 393–408, 2007.

[34] M. Alonso, S. Coll, J.-M. Martínez, V. Santonja, P. López, and J. Duato, "Power saving in regular interconnection networks", *Parallel computing*, vol. 36, no. 12, pp. 696–712, 2010.

[35] S Conner, S. Akioka, M. J. Irwin, and P. Raghavan, "Link shutdown opportunities during collective communications in 3-d torus nets", in *2007 IEEE International Parallel and Distributed Processing Symposium*, IEEE, 2007, pp. 1–8.

[36] C. Jackson and S. J. Hollis, "Skip-links: A dynamically reconfiguring topology for energy-efficient nocs", in *2010 International Symposium on System on Chip*, IEEE, 2010, pp. 49–54.

[37] R. Mullins, "Minimising dynamic power consumption in on-chip networks", in *2006 International Symposium on System-on-Chip*, IEEE, 2006, pp. 1–4.

[38] F. Worm, P. Ienne, P. Thiran, and G. De Micheli, "An adaptive low-power transmission scheme for on-chip networks", in *Proceedings of the 15th international symposium on System Synthesis*, 2002, pp. 92–100.

[39] M. Ali, M. Welzl, and S. Hellebrand, "A dynamic routing mechanism for network on chip", in *2005 NORCHIP*, IEEE, 2005, pp. 70–73.

[40] T. Simunic, S. P. Boyd, and P. Glynn, “Managing power consumption in networks on chips”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 1, pp. 96–107, 2004.

[41] M Sanaye-Pasand and H Khorashadi-Zadeh, “Transmission line fault detection & phase selection using ann”, in *International Conference on Power Systems Transients*, 2003, pp. 1–6.

[42] H. Khorashadi Zadeh, “An ann-based high impedance fault detection scheme: Design and implementation”, *International Journal of Emerging Electric Power Systems*, vol. 4, no. 2, 2005.

[43] A. Prodromou, A. Panteli, C. Nicopoulos, and Y. Sazeides, “Nocalert: An online and real-time fault detection mechanism for network-on-chip architectures”, in *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, IEEE, 2012, pp. 60–71.

[44] R. Parikh and V. Bertacco, “Udirec: Unified diagnosis and reconfiguration for frugal bypass of noc faults”, in *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, 2013, pp. 148–159.

[45] C. Iordanou, V. Soteriou, and K. Aisopos, “Hermes: Architecting a top-performing fault-tolerant routing algorithm for networks-on-chips”, in *2014 IEEE 32nd International Conference on Computer Design (ICCD)*, IEEE, 2014, pp. 424–431.

[46] R. Nakamura, Y. Sekiya, D. Miyamoto, K. Okada, and T. Ishihara, “Malicious host detection by imaging syn packets and a neural network”, in *2018 International Symposium on Networks, Computers and Communications (ISNCC)*, IEEE, 2018, pp. 1–4.

[47] R. Abou Khamis and A. Matrawy, “Evaluation of adversarial training on different types of neural networks in deep learning-based idss”, in *2020 international symposium on networks, computers and communications (ISNCC)*, IEEE, 2020, pp. 1–6.

[48] Y. Li, Y. Liu, M. Li, Y. Tian, B. Luo, and Q. Xu, “D2nn: A fine-grained dual modular redundancy framework for deep neural networks”, in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019, pp. 138–147.

[49] A. Raghunathan, J. Steinhardt, and P. Liang, “Certified defenses against adversarial examples”, *arXiv preprint arXiv:1801.09344*, 2018.

[50] T.-W. Weng, P. Zhao, S. Liu, P.-Y. Chen, X. Lin, and L. Daniel, “Towards certificated model robustness against weight perturbations”, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 6356–6363.

[51] F. Ponzina, M. Peon-Quiros, A. Burg, and D. Atienza, “E 2 cnns: Ensembles of convolutional neural networks to improve robustness against memory errors in edge-computing devices”, *IEEE Transactions on Computers*, vol. 70, no. 8, pp. 1199–1212, 2021.

[52] Y.-L. Tsai, C.-Y. Hsu, C.-M. Yu, and P.-Y. Chen, “Formalizing generalization and adversarial robustness of neural networks to weight perturbations”, *Advances in Neural Information Processing Systems*, vol. 34, pp. 19 692–19 704, 2021.

[53] P. Arcaini, A. Bombarda, S. Bonfanti, and A. Gargantini, “Roby: A tool for robustness analysis of neural network classifiers”, in *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*, IEEE, 2021, pp. 442–447.

[54] N. Raviv, P. Upadhyaya, S. Jain, J. Bruck, and A Jiang, "Coded deep neural networks for robust neural computation", in *Proceedings of the Non Volatile Memories Workshop (NVMW), San Diego, CA, USA*, 2020, pp. 8–10.

[55] N. Raviv, A. Kelley, M. Guo, and Y. Vorobeychik, "Enhancing robustness of neural networks through fourier stabilization", in *International Conference on Machine Learning*, PMLR, 2021, pp. 8880–8889.

[56] H. Lim, S.-D. Roh, S. Park, and K.-S. Chung, "Robustness-aware filter pruning for robust neural networks against adversarial attacks", in *2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP)*, IEEE, 2021, pp. 1–6.

[57] B. Zhang, T. Cai, Z. Lu, D. He, and L. Wang, "Towards certifying l-infinity robustness using neural networks with l-inf-dist neurons", in *International Conference on Machine Learning*, PMLR, 2021, pp. 12 368–12 379.

[58] H. Hossain, M. Ahmed, A. Al-Nayeem, T. Z. Islam, and M. M. Akbar, "Gpnocsim-a general purpose simulator for network-on-chip", in *2007 International Conference on Information and Communication Technology*, IEEE, 2007, pp. 254–257.

[59] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: A power-performance simulator for interconnection networks", in *35th Annual IEEE/ACM International Symposium on Microarchitecture, 2002.(MICRO-35). Proceedings.*, IEEE, 2002, pp. 294–305.

[60] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: Characterization and architectural implications", in *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, 2008, pp. 72–81.

[61] Y. Hoskote, S. Vangal, S. Dighe, N. Borkar, and S. Borkar, "Teraflops prototype processor with 80 cores", in *2007 IEEE Hot Chips 19 Symposium (HCS)*, IEEE, 2007, pp. 1–15.

[62] W.-C. Cheng and M. Pedram, "Low power techniques for address encoding and memory allocation", in *Proceedings of the 2001 Asia and South Pacific Design Automation Conference*, 2001, pp. 245–250.

[63] G. Southern and J. Renau, "Analysis of parsec workload scalability", in *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, IEEE, 2016, pp. 133–142.

[64] A. G. Savva, T. Theocharides, C. Nicopoulos, *et al.*, "A design space exploration framework for ann-based fault detection in hardware systems", *Journal of Electrical and Computer Engineering*, vol. 2017, 2017.

[65] M. Clark, A. Kodi, R. Bunescu, and A. Louri, "Lead: Learning-enabled energy-aware dynamic voltage/frequency scaling in nocs", in *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1–6.

[66] R. B. Tonetto, D. M. Cardoso, M. Brandalero, L. Agostini, G. L. Nazar, J. R. Azambuja, and A. C. S. Beck, "A knapsack methodology for hardware-based dmr protection against soft errors in superscalar out-of-order processors", in *2019 IFIP/IEEE 27th International Conference on Very Large Scale Integration (VLSI-SoC)*, IEEE, 2019, pp. 287–292.

[67] A. Savva, *Parsec - data*, Accessed: 10 June, 2024. [Online]. Available: `https://www.github.com/Andsa1/PARSEC-Data`.

[68] B. Mburano, W. Si, and W. X. Zheng, "A comparative study on the variants of r metric for network robustness", in *2021 International Symposium on Networks, Computers and Communications (ISNCC)*, IEEE, 2021, pp. 1–6.