Hindawi
Journal of Electrical and Computer Engineering
Volume 2017, Article ID 9361493, 12 pages
https://doi.org/10.1155/2017/9361493

Research Article

A Design Space Exploration Framework for ANN-Based Fault Detection in Hardware Systems

Andreas G. Savva, Theocharis Theocharides, and Chrysostomos Nicopoulos

University of Cyprus, Nicosia, Cyprus

Correspondence should be addressed to Andreas G. Savva; andsavv_1981@yahoo.com

Received 28 July 2017; Accepted 29 October 2017; Published 3 December 2017

Academic Editor: Ping Feng Pai

This work presents a design exploration framework for developing a high level Artificial Neural Network (ANN) for fault detection in hardware systems. 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 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 interrouter link faults. Simulation results with various synthetic traffic models show that the proposed work can detect up to 96–99% of interrouter link faults with a delay less than 60 cycles. Added to this, the size of the ANN is kept relatively small and they can be implemented in hardware easily. Synthesis results indicate an estimated amount of 0.0523 mW power consumption per neuron for the implemented ANN when computing a complete cycle.

1. Introduction

This work 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. 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 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. The size of the ANNs is kept relatively small and the ANN complexity is minimized, and the monitoring of the entire hardware system is done in parallel and independently by each ANN. Moreover, ANNs can easily be designed and implemented in hardware and their size for this work remains relatively small.

The major problem which affects the reliability in hardware systems/networks 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 remains one of the top challenges and the development of an intelligent fault detection scheme is needed. For the purposes of evaluating the ANNs used for fault detection in this work, a case study scenario based on NoCs is used. In future on-chip generations, there will be a huge increase in faults [2, 3]. According to ITRS, in the near future, the manufacturing defect rate will reach approximately up to 1000 defects/m^2 [3]. 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 [4, 5].

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 [6–8]. Most of them present low level prediction mechanisms.

This work focuses on the analysis of the development of an exploration 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 the current work, and with the use of a NoC case study scenario, simulation results in 2D mesh topologies show good detection results (up to 96–99% correct interrouter link fault detection under the use of synthetic traffic models), with very low additional accepted delays and costs. The additional hardware overhead from the use of ANNs is very small. These results show that ANNs can be successfully used for fault detection in different hardware systems.

This work is organized as follows. Section 2 discusses the related work. Section 3 introduces and analyzes the design of the framework based on different parameters. Section 4 presents the case study simulated framework with the result analysis and Section 5 gives conclusions for future research.

2. Background and Related Work

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 nonlinear 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 [9–12]. Based on these, ANNs offer high prediction capabilities. ANNs also have been successfully applied in various real life scenarios which include learning systems [13], neuroscience [14], and engineering [15]. 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 interrouter link fault detection. Integrated hardware-based ANNs are used, and, based on the appropriate ANN training and the received link utilization data in discrete interval times, intelligently detect which router might present fault. ANN size remains relatively small and can be efficiently designed in hardware.

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 [12]. According to Shamishi et al. [16], 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 et al. [9, 11] 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 [17].

Next subsection presents some of the previous work done for fault detection in Networks on Chip since in the future ANNs can be used for this purpose.

2.2. Fault Detection in NoCs Related Work. Sanaye et al. [18] 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 implements fault detection, classification, and fault phase selection for transmission lines. The authors in [19] have recently presented a new method for detection of 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, the authors in [20] 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 wrong outputs based on the current inputs (microchecker models in hardware). In [21], the 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 [22], the 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 preconfigured paths in case of faults.

The work in [17] 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. This work uses ANNs for detection

Table 1: Decision steps needed to be taken 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.
4	Fault detection	Study different scenarios for fault detection (study two scenarios: first, the detection is made just to show which partition will present fault; second, the detection shows which router in the partition will present fault).

Table 2: 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

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, and latency. Motivated from previous works and the ideas in [17], this work creates as a case study scenario an intelligent framework for interrouter link fault detection in NoCs and explains how integrated small-sized hardware-based ANNs can be used for this purpose.

3. Framework Methodology

3.1. Framework Development. Many different decision steps must be taken in order to choose the optimal parameters for developing correctly the exploration framework. Table 1 shows the decision steps needed for the general framework from the ANN perspective, while Table 2 presents the decision steps needed for the simulator. ANNs can be used in any hardware system. In order to evaluate the ANNs, a NoC is used as a case study. 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 and the simulation framework have to be analyzed. This work is focused firstly on the simulator for the collection of training data for the ANNs and also for the simulation results (detection versus 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 sections. 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 interrouter link faults. 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.

3.2. Topology Exploration: ANN Scalability. NoC is a packet-switched network which connects all the functional units on the chip providing communication infrastructure in order to configure many-core systems on chip and is going to be used as a case study scenario in order to evaluate the whole framework.

gpNoCsim simulator (General Purpose Simulator for Network-on-Chip Architectures) is used in order to create the network on-chip topology.

gpNoCsim is an open-source, component based simulation framework for Networks-on-Chip architectures that is developed in Java language. This framework is built upon the object oriented modular design of the NoC architecture components [23].

We assumed an 8×8 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 network, on top of the NoC topology. Different NoC partitions can be implemented and analyzed in order to choose which one is appropriate for the fault detection framework. NoC topology and ANNs are presented and analyzed in detail in Section 4.

3.3. ANN Mechanism Overview and Training. The ANN mechanism can be considered as an independent processing

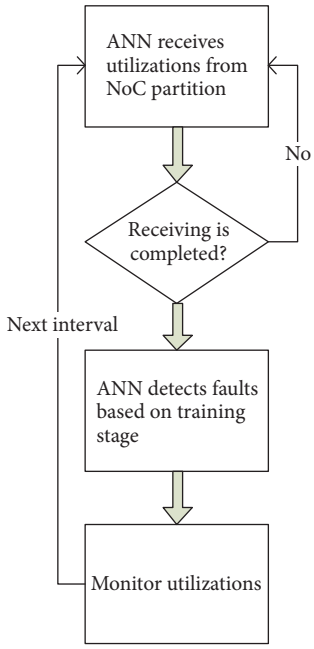

Figure 1: Overview of the ANN detection process.

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 for which it is responsible, and these values are then processed by the ANN in order to make the detection.

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 presented in Figure 1.

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

4. Framework Evaluation Case Study

4.1. Scalability: ANN Partitioning. For scalability purposes, partitioning the case study NoC into smaller regions (e.g., an 8×8 NoC into four 4×4 regions or four 4×5 regions or 5×4 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 and easy to implement in hardware. The ANN mechanism can be considered as a different independent network, on the top of the topology.

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 2 shows the different ANN partitions studied (4×4, 5×4, 4×5).

Figure 3 presents the resulting ROC curves for 4×4, 4×5, and 5×4 topologies in the case of router fault detection. Different ROC curves were created with the use of different traffic patterns. Partition 4×5 is more efficient since it produces better resulting ROC curves for different traffic patterns compared with 4×4 and 5×4 partitions (partition 4×5 ROC curves present slightly better results compared with the results of the 5×4 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 4×5 partitions, for the case of detecting which router in the region will present fault.

4.2. 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. This makes it easy to implement in hardware [17]. 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, and Neighbor [17]), using an offline and back-propagation ANN training algorithm [1]. For the purposes of this article, 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 for detecting which partition will present a fault. For the second scenario, the ANN is responsible for detecting which router in the partition will present fault. For these partitions, in the case of detecting fault in the whole ANN, the ANN should only have one output neuron. For the case of detecting which router in the region will more likely present fault, the ANN should have 20 output neurons, one for each region router in the cases of 5×4, 4×5 NoC partitions and 16 output neurons for the 4×4 NoC partition case.

Figure 4 presents the ANN architecture used in the two different case scenarios for this article. Figure 4(a) shows the ANN architecture for the case of detecting the partition which will have a faulty router. Figure 4(b) shows the ANN architecture for the case of detecting which router in the ANN region will present fault. The difference is presented in the output layer. In the first case, only one output neuron is needed which will show which ANN region might have fault. In the second case, the number of output neurons depends on the number of the routers included in the ANN region which is simulated.

4.3. 4×5 ANN Base Model. An ANN which is responsible to monitor a 4×5 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 Figure 5.

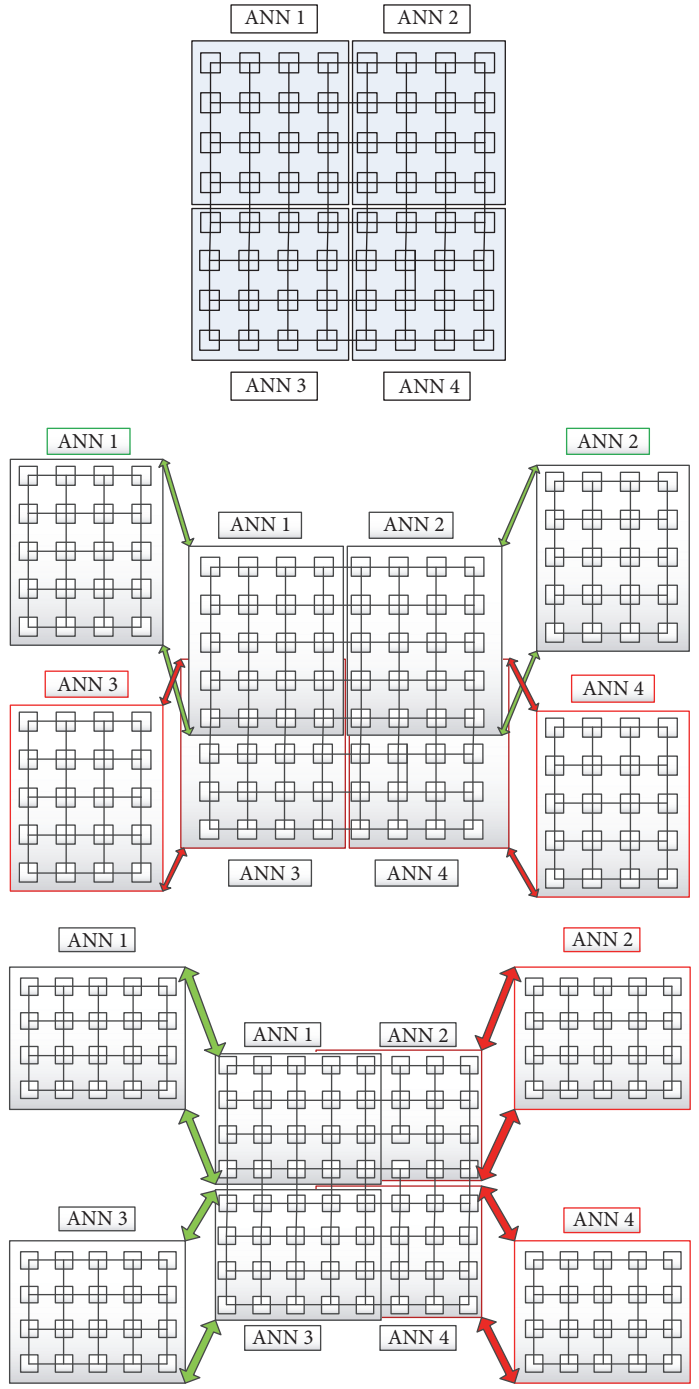

Figure 2: Different ANN partitions: 4 × 4 partitions, 5 × 4 partitions, and 4 × 5 partitions.

4.4. ANN Parameters and 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 NoC topology is partitioned into smaller regions and a base ANN is assigned to be responsible for each region. We assume that the ANNs are different networks on top of the NoC topology.

The ANN receives the link utilization values from all the router ports of the partition which it 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 in order to intelligently detect which ports will present fault. A neural network can be implemented in hardware by using multiplier-accumulator (MAC) units and a lookup table (LUT) as activation function [17].

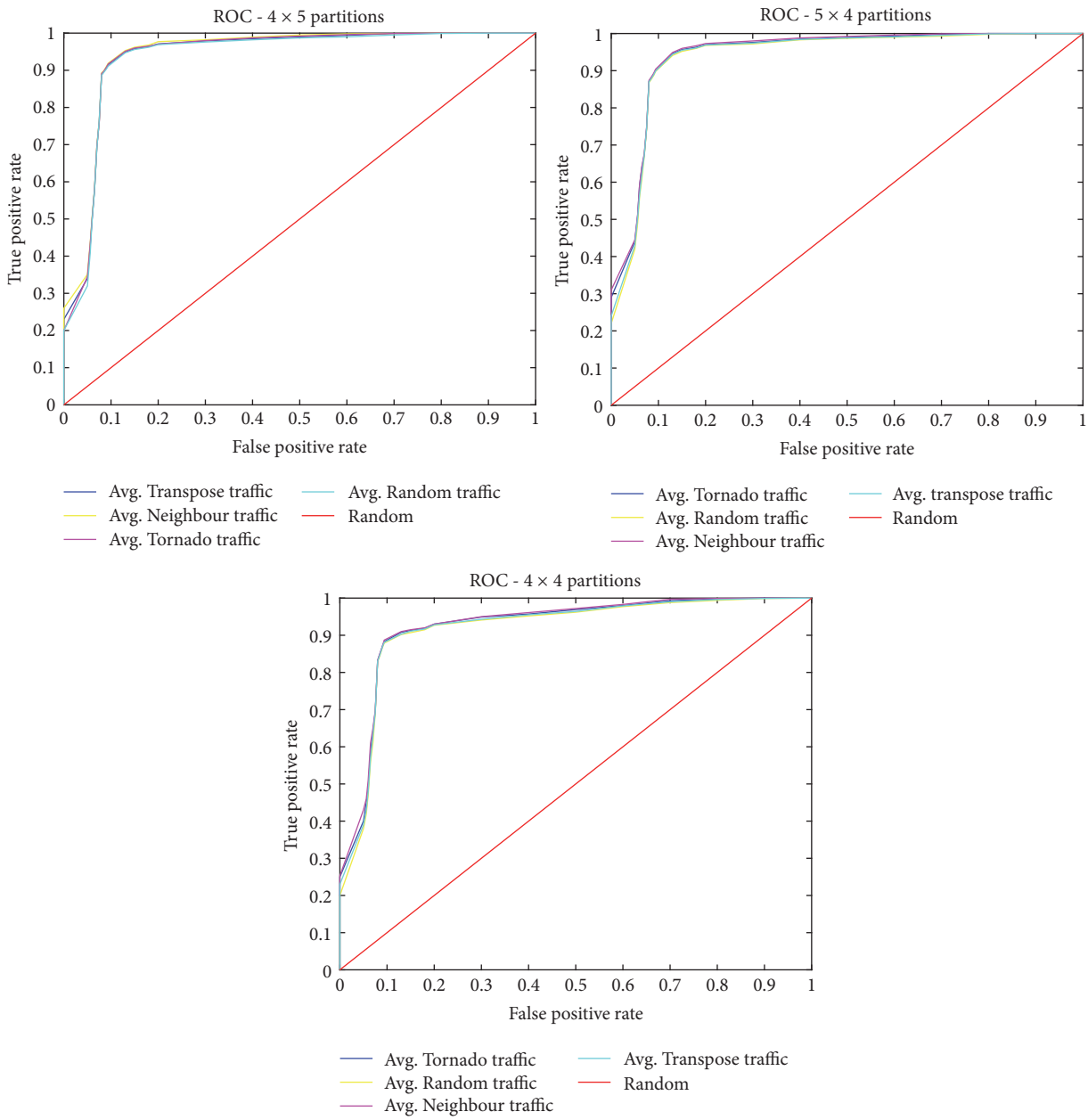

Figure 3: Resulting ROC curves for 4 × 4, 4 × 5, and 5 × 4 partitions.

The ANN training stage can be performed offline when the NoC is not used and the training weights can be stored in SRAM based LUTs for fast and online reconfiguration of the network. The network is trained with the use of application traffic patterns, offline, and any ANN training algorithm can be used. In our experiments, we used synthetic traffic patterns and the MATLAB ANN toolbox; the weight values were fed to the simulator as inputs, where the actual detection was then implemented and simulated.

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 and faults. The number of hidden layer neurons can be selected to be half of the summation of the input and output data [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 6 shows the resulting ROC graphs for different number of neurons in the hidden layer. Table 3 presents the configuration parameters used for each resulting ROC graph.

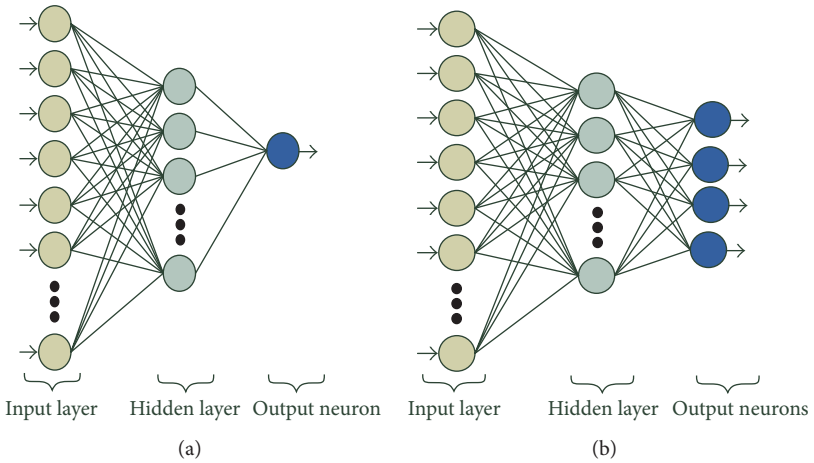

Figure 4: ANN architecture with the input, hidden, and output layers for detection of (a) fault in whole ANN and (b) fault in router.

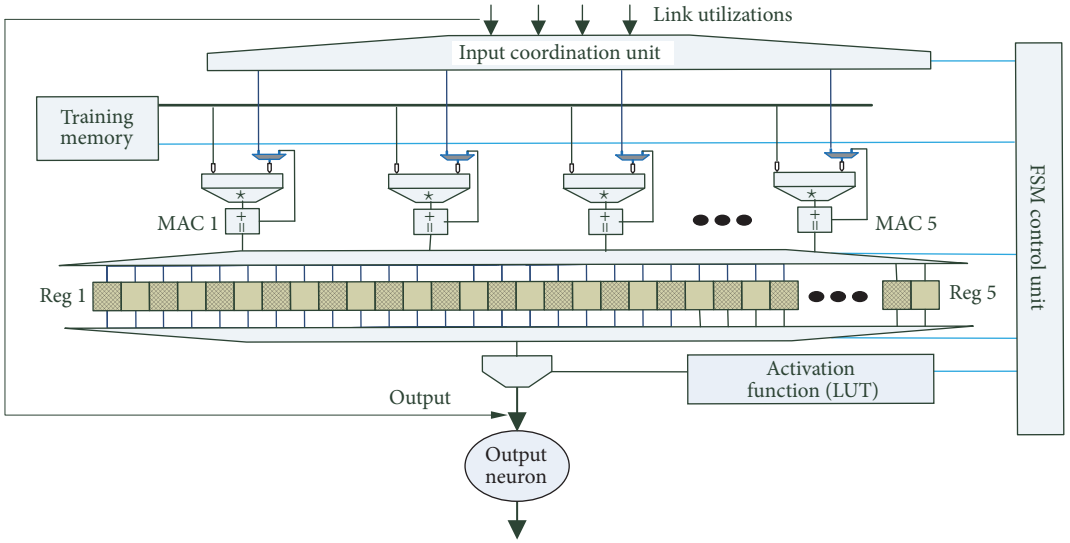

Figure 5: ANN hardware architecture.

Table 3: ANN configurations with different number of neurons in the hidden layer.

ANN configuration	Number of neurons in hidden layer
ANN configuration 1	17 neurons in hidden layer
ANN configuration 2	21 neurons in hidden layer
ANN configuration 3	20 neurons in hidden layer
ANN configuration 4	18 neurons in hidden layer
ANN configuration 5	19 neurons in hidden layer

Based on Figure 6, 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.

4.5. 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, and 100 cycle time intervals were studied and based on the experimental results an optimal time interval is selected. Based on the results of Figure 7, 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 4 presents the configuration parameters used for each resulting ROC graph in Figure 8.

Figure 8 presents the resulting curves. The best results comparing the ROC curves for the different sentinel values

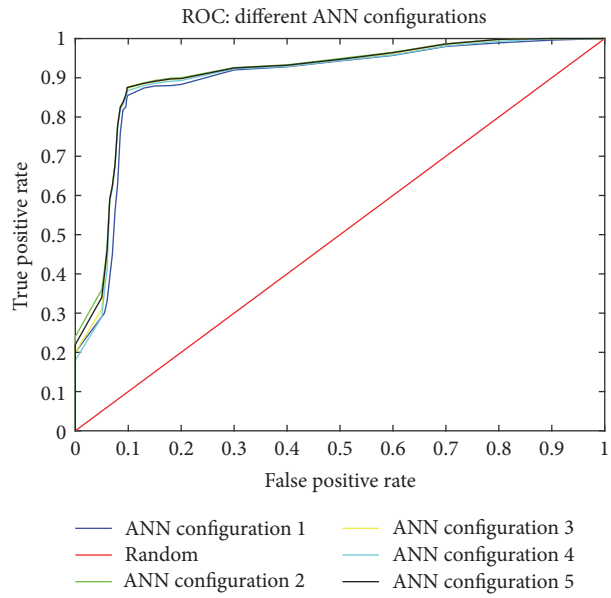

Figure 6: ROC graphs for different neurons in hidden layer.

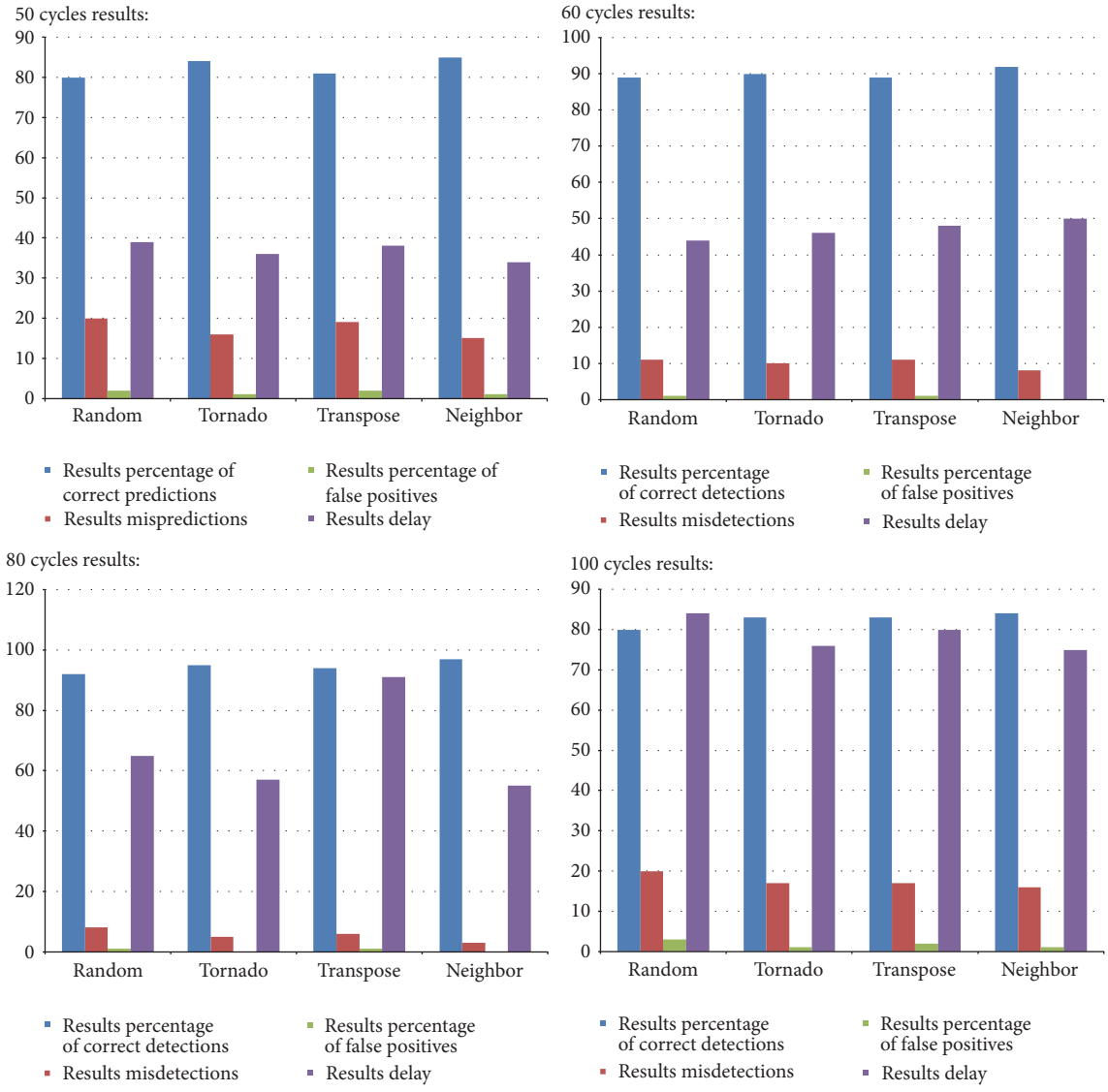

Figure 7: Results for different cycle time intervals.

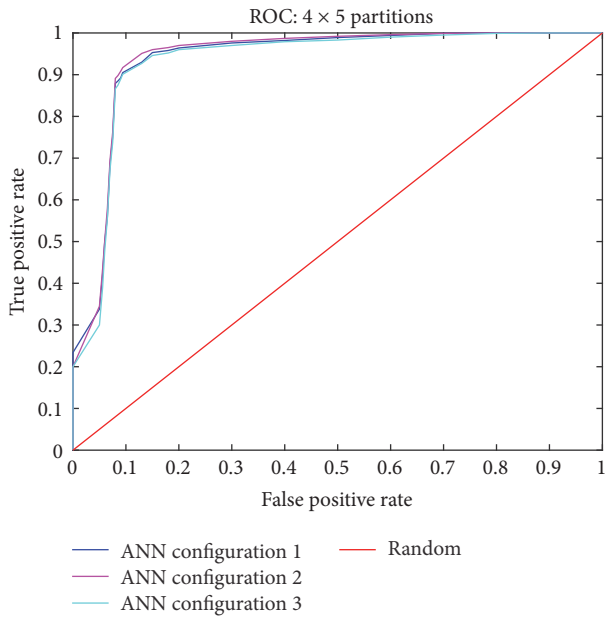

Figure 8: Resulting ROC graphs for different sentinel values.

Table 4: ANN configurations for different sentinel values.

ANN configuration	Number of neurons in hidden layer/sentinel value
ANN configuration 1	19 neurons in hidden layer, sentinel value 70
ANN configuration 2	19 neurons in hidden layer, sentinel value 50
ANN configuration 3	19 neurons in hidden layer, sentinel value 20

are shown in the case of 50 cycles sentinel value. Based on the results, an efficient value to work with is 50 cycles.

4.6. Topology Exploration Setup: Adaptability in Various Hardware Configurations. In order to evaluate the developed ANNs, various hardware configurations can be used. For the purposes of this work and for the NoC case study scenario, gpNoCsim simulator is used. This simulator uses an adaptive routing algorithm based on XY algorithm with a modified turn model with virtual channel support and 4-stage pipelined router operation [23, 24]. It also supports 64-bit flit width with worm-whole flow control, 2 virtual channels per link with buffer depth of 5 flits per virtual channel, and different traffic models.

Simulation experiments were ran over 8 × 8 NoC topologies, case study scenario, where partitions into four regions of 4 × 5 routers are created. Each ANN-based model is responsible for one of these regions. Simulations are done over 200,000 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 fault ports.

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 to 0.3 flits/node/cycle in steps of 0.05 flits/node/cycle from low to high. Added to this, 4 different traffic patterns are studied in three different scenarios of fault injections (at cycles: 0, 32000, and 64000). Three injection rates, multiplied by three injection times, multiplied by four traffic patterns, multiplied by 223 links, equals 8028 different fault injection simulations. Simulation results are presented in the next section.

4.7. Framework Evaluation. The number of corrected detection instances 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 detection is measured. In order to succeed in 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 of 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 9 summarizes the comparison when targeting the case study scenario of an 8 × 8 mesh NoC. For all the traffic patterns, the number of undetected/unexpected faults, in comparison to the correct detection, is very low (less than 4). Neighbor and Tornado traffic patterns show better results concerning the percentage of correct detection (98-99% and 97–99%, resp.) and less delay (55 and 58 cycles, resp.) compared to Transpose and Random traffic patterns (96-97%). Added to this, simulations show that the misdetection for the cases of Neighbor, Tornado, and Transpose traffic patterns are very low (less than 2). Random traffic pattern presents more miss detection but this is acceptable compared with the high percentage of correct detection.

4.8. ANN Costs and Power Consumption. For a 64-input ANN, a hyperbolic tangent activation function, and a single threshold subtractor (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

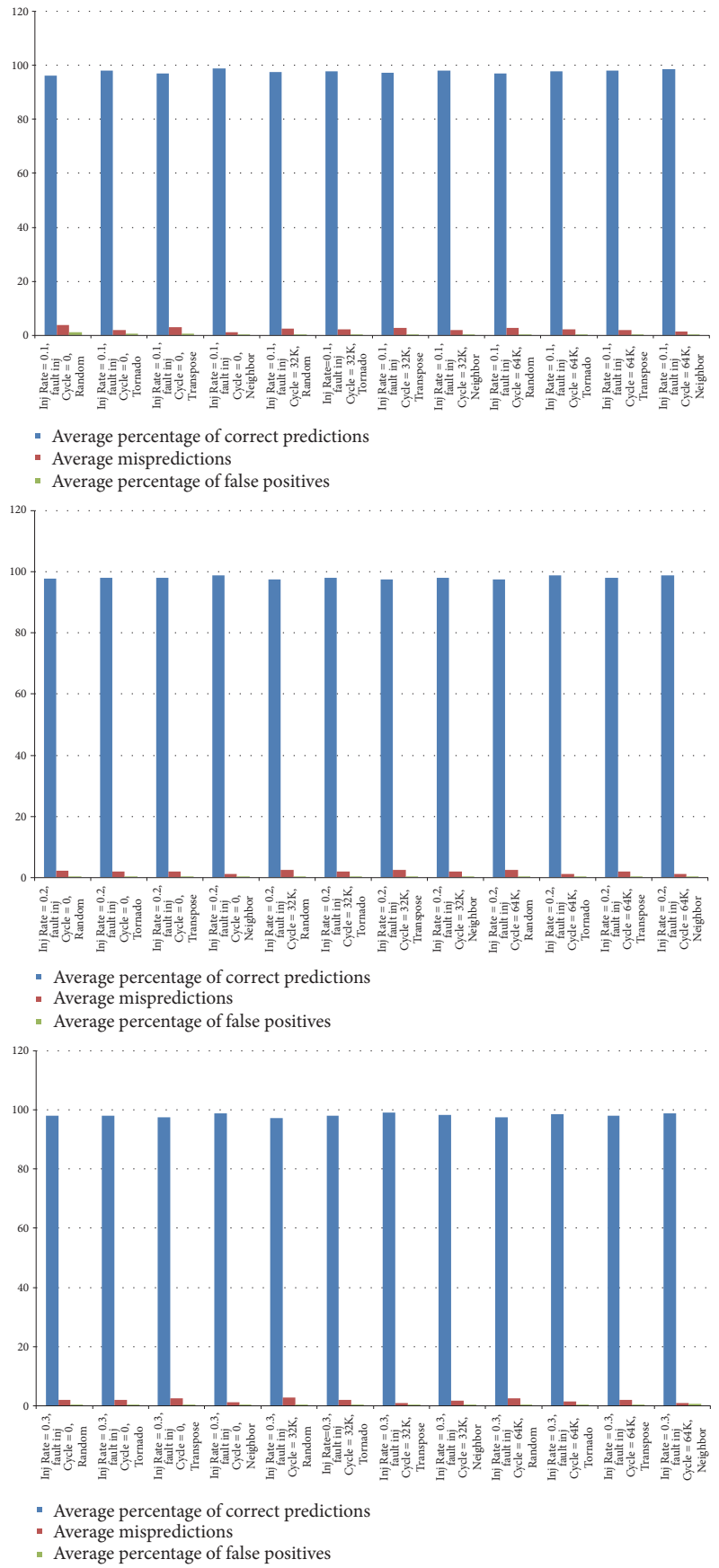

Figure 9: Detection results for four different traffic patterns (Random, Tornado, Transpose, and Neighbor) with three different injection rates (0.1–0.3) and three different fault injection cycles (0, 32 K, and 64 K).

frequency of 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 16 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 need 19 cycles in order to reach the next layer of neurons. So, 19 cycles $*$ 0.00275 mW = 0.0523 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 reused so there will be no need to compensate for extra area.

5. Conclusions

This work presents a design exploration framework for fault detection in hardware systems with the use of high level ANNs. It analyzes, explains, and evaluates 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 interrouter 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, 4×5 ANNs with 19 neurons in hidden layer and four output neurons were chosen for this work. Based on experiments on 8×8 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 96–99% under synthetic 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.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: a tutorial," *IEEE Computational Science & Engineering*, vol. 29, no. 3, pp. 31–44, 1996.

[2] R. A. Shafik, J. Mathew, and D. K. Pradhan, "Introduction to energy-efficient fault-tolerant systems," *Energy-Efficient Fault-Tolerant Systems*, pp. 1–10, 2014.

[3] ITRS, http://www.itrs.net/.

[4] S. Murali, T. Theocharides, N. Vijaykrishnan, M. J. 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.

[5] C. Grecu, A. Ivanov, R. Saleh, E. S. Sogomonyan, and P. P. Pande, "On-line fault detection and location for NoC interconnects," in *Proceedings of the IOLTS 2006: 12th IEEE International On-Line Testing Symposium*, pp. 145–150, Italy, July 2006.

[6] 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.

[7] J. Liu, J. Harkin, Y. Li, and L. Maguire, "Online fault detection for NoC interconnect," *IEEE Computer Architecture Letters*, 2013.

[8] C. Grecu, P. Pande, A. Ivanov, and R. Saleh, "BIST for network-on-chip interconnect infrastructures," in *Proceedings of the 24th IEEE VLSI Test Symposium*, pp. 30–35, USA, May 2006.

[9] M. Jahirul, P. Brooks et al., "An ANN Model for Predicting Biodiesel Kinetic Viscosity as a Function of Temperature and Chemical Compositions," in *Proceedings of the Proc. 20th International Congress on Modeling and Simulation*, Australia, 2013.

[10] V. Pacelli, V. Bevilacqua, and M. Azzollini, "An artificial neural network model to forecast exchange rates," *Journal of Intelligent Learning Systems and Applications*, vol. 3, no. 2, pp. 57–69, 2011.

[11] K. Suzuki, *Artificial Neural Networks - Methodological Advances and Biomedical Applications*, InTech, 2011.

[12] Z. Khan, T. Alin, and A. Hussain, "Price Prediction of Share Market using ANN," *In Proc. IJCA Journal, Studies in Informatic and Control*, vol. 7, no. 3, pp. 111–120, 1998.

[13] S. Carillo, J. Harkin, L. McDaid et al., "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.

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

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

[16] M. Shamishi et al., "Using matlab to develop ANN methods for predicting global solar radiation," *Engineering Education and Research using Matlab*, pp. 978-953, 2011.

[17] A. G. Savva, T. Theocharides, and V. Soteriou, "Intelligent on/off link management for on-chip networks," in *Proceedings of the 2011 IEEE Computer Society Annual Symposium on VLSI, ISVLSI 2011*, pp. 343-344, India, July 2011.

[18] M. Sanaye et al., "Transmission Line Fault Detection and Phase Selection using ANN," in *Proceedings of the International Conference on Power Systems Transients, IPST*, 2003.

[19] H. Zadeh, "An ANN-based high impedance fault detection scheme: design and implementation," *International Journal of Emerging Electric Power Systems*, vol. 4, no. 2, 2005.

[20] A. Prodromou, A. Panteli, C. Nicopoulos, and Y. Sazeides, "NoCAlert: An on-line and real-time fault detection mechanism for network-on-chip architectures," in *Proceedings of the 2012 IEEE/ACM 45th International Symposium on Microarchitecture, MICRO 2012*, pp. 60–71, Canada, December 2012.

[21] 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, MICRO 2013*, pp. 148–159, USA, December 2013.

[22] C. Iordanou, V. Soteriou, and K. Aisopos, "Hermes: Architecting a top-performing fault-tolerant routing algorithm for Networks-on-Chips," in *Proceedings of the 32nd IEEE International Conference on Computer Design, ICCD 2014*, pp. 424–431, Republic of Korea, October 2014.

[23] H. Hossain, M. Ahmed, A. Al-Nayeem, T. Z. Islam, and M. M. Akbar, "gpNoCsim - A general purpose simulator for network-on-chip," in *Proceedings of the ICICT 2007: International Conference on Information and Communication Technology*, pp. 254–257, Bangladesh, March 2007.

[24] S. Jovanovic, C. Tanougast, S. Weber, and C. Bobda, "A new deadlock-free fault-tolerant routing algorithm for noc interconnections," in *Proceedings of the FPL 09: 19th International Conference on Field Programmable Logic and Applications*, pp. 326–331, Czech Republic, September 2009.