TSA-NoC: Learning-Based Threat Detection and Mitigation for Secure Network-on-Chip Architecture

Ke Wang, Hao Zheng, and Ahmed Louri
George Washington University

***Abstract*—Networks-on-chip (NoCs) are playing a critical role in modern multicore architecture, and NoC security has become a major concern. Maliciously implanted hardware Trojans (HTs) inject faults into on-chip communications that saturate the network, resulting in the leakage of sensitive data via side channels and significant performance degradation. While existing techniques protect NoCs by detecting and isolating HT-infected components, they inevitably incur occasional inaccurate detection with considerable network latency and power overheads. We propose TSA-NoC, a learning-based design framework for secure and efficient on-chip communication. The proposed TSA-NoC uses an artificial neural network for runtime HT-detection with higher accuracy. Furthermore, we propose a deep-reinforcement-learning-based adaptive routing design for HT mitigation with the aim of minimizing network latency and maximizing energy efficiency. Simulation results show that TSA-NoC achieves up to 97% HT-detection accuracy, 70% improved energy efficiency, and 29% reduced network latency as compared to state-of-the-art HT-mitigation techniques.**

As technology scales, modern multiprocessors have pushed for a paradigm shift from computation-centric to communication-centric systems. Networks-on-Chip (NoCs) are becoming increasingly critical yet vulnerable to various security threats,[1–4] especially to hardware trojans (HTs).[4] Maliciously implanted HTs inject transient faults in transmitted flits/packets, causing misrouting and unnecessary packet re-transmissions.

Digital Object Identifier 10.1109/MM.2020.3003576
Date of publication 19 June 2020; date of current version 1 September 2020.

Published by the IEEE Computer Society

These retransmissions consume massive NoC resources (e.g., link and buffer) and saturate the transmission channels, resulting in data leakage, significant performance degradation, and even a denial-of-service.[2,3]

Significant research has been devoted to securing NoCs against HT-based attacks.[5,6] A majority of these countermeasures mitigate HTs by detecting and isolating the HT-infected NoC components. In the HT-detection aspect, the existing works use fault history logging (FHL),[7] runtime threshold monitoring (RTM) on link-error/packet-injection rates, or built-in self-testing hardware.[6] These techniques monitor fault-related NoC attributes (e.g., temperature and buffer/link utilization) at runtime and label the corresponding component as HT-infected if any attribute value exceeds its corresponding manually set threshold. However, since massive NoC attributes are correlated with transient faults and interact with each other,[8] designing the thresholds for HT detection is complicated. These thresholds, if selected carelessly, can cause false/misdetection, additional power consumption, and increased network latency.

In the HT-isolation aspect, conventional solutions use regional routing algorithms[2,3] to isolate the HT-infected components. Unfortunately, these techniques limit the network throughput, forbid communication via certain channels, and detour the packets to avoid infected regions. This inevitably increases network latency. Therefore, the challenge is to design a secure architecture that promptly and accurately detects and isolates HTs with minimal performance loss.

In this article, we propose TSA-NoC, a learning-based, high-performance, and energy-efficient NoC design for secured on-chip communication. In TSA-NoC, we enhance the router architecture with a learning-based per-router HT-detection (DetectANN) module, bypass channel, and a SmartRoute controller for HT-isolation. The major contributions of this article include the following.

- *Improved HT detection using DetectANN:* DetectANN uses an artificial neural network (ANN) to automatically identify HT-injected faults by recognizing abnormal network behaviors (e.g., unexpected high error rate) and improve the accuracy of HT detection.
- *Improved HT-isolation using SmartRoute:* After HT detection, the routers are dynamically categorized into HT-free and HT-infected routers. A low-cost bypass channel using simple switch logic is proposed to bypass the HT-infected routers while maintaining network connectivity. To balance traffic loads among low-throughput bypass channels and high-throughput routers and improve the overall network performance, SmartRoute controller uses DRL to handle diverse traffic patterns by dynamically applying the most suitable routing algorithms thus minimizing network latency and power consumption.

We evaluate the performance of the proposed TSA-NoC architecture with PARSEC benchmark. Simulation results show that TSA-NoC provides enhanced HT-detection accuracy and improved HT mitigation with reduced power consumption and network latency as compared to conventional HT-mitigation techniques.

BACKGROUND AND MOTIVATION

Transient Faults in NoC

Transient faults may manifest during any stage of transmission[8] in NoCs. To mitigate these faults in typical NoC architecture, each flit is encoded with error correction codes (ECCs) before being propagated to the downstream router. At each hop of transmission, the ECC decoder examines the correctness of each flit and initiates a negative-acknowledgment signal for data retransmission if an error occurs.

HT Attacks on NoCs

HTs have been shown to infect NoC links[9] and router microarchitectures.[5] Typically, HTs are inserted in the layout during the fabrication phase of the IC design cycle. After being implanted, they usually remain dormant and slip past hardware diagnosis until they are activated by attackers. Once activated, they inject transient faults by forcing bit-flips in links or disrupting ECC in routers. These faults can lead to massive retransmission traffic, back pressures, and network saturation.

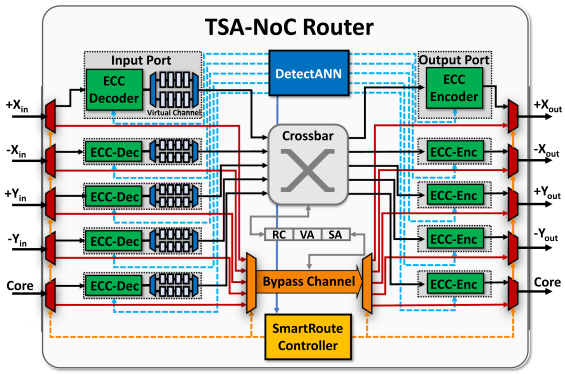

Figure 1. Proposed TSA-NoC router architecture. The router comprises a DetectANN design for HT detection, a bypass channel for HT isolation, and a SmartRoute controller. Solid and dashed arrows represent data paths and control signals, respectively.

Conventional HT-Mitigation Techniques

A majority of the existing HT-mitigation techniques use a detect–isolate approach.[2,3,6,7] For HT detection, built-in diagnosis hardware[6] periodically checks the correctness of the circuitry's logic operations, which inevitably stalls the application execution to retain HT-detection accuracy. To mitigate this limitation, runtime software-based mechanisms such as FHL[7] and RTM are proposed to detect suspicious HT-infected components by dynamically capturing abnormalities in NoC behavior. FHL records the pattern of transient faults in all the transmitted packets to identify HT-infected channels. Similarly, RTM monitors the retransmission rate and marks out suspicious HT-infected components that exceed a certain retransmission rate. However, both techniques rely on manually-set thresholds to identify suspicious NoC behavior, thus, often leading to false HT detection.

For HT isolation, SurfNoC[3] eliminates the transmissions between high-security and low-security domains by alternatively reserving and scheduling transmission channels in each dimension exclusively for a specific domain. NIBR[2] partitions the virtual channels of each router to transmit data flows exclusively with high-security demands. However, the static network partition results in poor network utilization, thus, affecting network performance.

PROPOSED TSA-NoC DESIGN

We propose TSA-NoC, a learning-based HT-detection and mitigation framework for secure

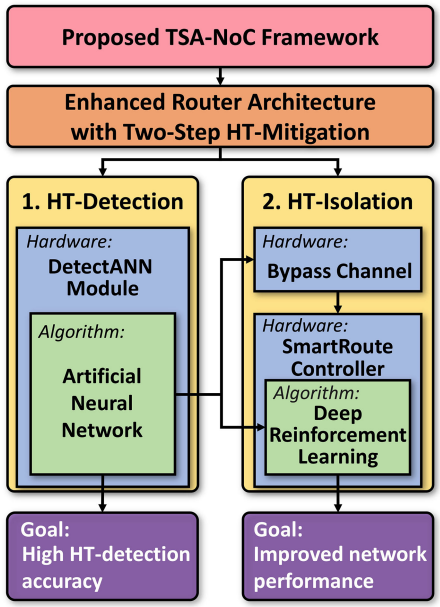

Figure 2. Overview of the proposed TSA-NoC design.

NoC architecture. In TSA-NoC, we propose an enhanced router architecture, as shown in Figure 1, consisting of an improved HT-detection design using an ANN (DetectANN), a bypass channel, and an enhanced HT-mitigation mechanism (SmartRoute) to improve network performance with DRL.

The overview of the proposed TSA-NoC framework is shown in Figure 2. DetectANN monitors network attributes, learns from runtime network activities, and automatically identifies HT-infected components by recognizing abnormal network behaviors (e.g., high error rate and retransmission). Based on the HT-detection results from DetectANN, SmartRoute controller categorizes packets into ones whose source and destination nodes are HT-free (high-security packets) and ones whose source and/or destination nodes are HT-infected (low-security packets). For high-security packets, all the components in the transmission path should be HT-free for security, while the transmission paths of the latter packets are allowed to contain HT-infected nodes. To isolate the HT-infected routers to protect the high-security packets, a bypass channel is proposed to bypass the HT-infected routers while maintaining network connectivity. Since the simple switch logic of the bypass channel inevitably degrades the network

throughput, a routing mechanism that avoids transmitting intense traffic through bypass channels should be applied. Moreover, to better utilize network resources, especially the isolated routers, a different routing algorithm may be needed for the low-security packets. To this end, we propose SmartRoute, which proactively selects the most suitable routing algorithm to balance traffic-loads in the low-throughput bypass channel and high-throughput routers, respectively, and improve overall network performance.

Learning-Based Runtime HT-Detection Using DetectANN

We implement a per-router DetectANN to perform runtime HT-detection with improved accuracy with minimized timing and power overheads. Unlike the static thresholds used by FHL and RTM, DetectANN eschews the human engineering, monitors NoC attributes, and automatically detects HTs by learning how to accurately recognize the abnormal behavior(s) of the local router through complex and interrelated NoC attributes. Since HTs are hard to detect when dormant, to identify activated HTs in a timely manner while reducing the computational overhead of DetectANN, HT-detection is performed iteratively.

Construction of DetectANN. DetectANN is a fully connected ANN with an input layer, a middle layer, and an output layer. Previous works have shown that some system attributes are highly correlated with transient faults in NoCs.[8] In TSA-NoC, we explore 12 fault-related NoC attributes as inputs, including buffer utilization (number of occupied virtual channels) for each input port (+x, −x, +y, −y, and local core), link utilization (value of input-flits per cycle) for each input port (+x, −x, +y, −y, and local core), local operation temperature, and the previous transient error rate in the last epoch. The middle layer utilizes all the attribute values and maps them to the classification of whether the router is HT-infected. As HT-detection is a binary classification problem, a single-hidden-layer construction can mitigate overfitting and is sufficient to deliver desired accuracy. For optimized detection accuracy and computational/storage overhead, we implement 30 neurons in the single hidden-layer (detailed discussion is in the "Evaluation and Analysis" section). The middle layer uses Sigmoid activation function. For each neuron j in the middle layer, the output $y_j = \text{sigmoid}(\Sigma_{i=0}^{12} x_i * w_i)$. The output layer indicates the binary classification result: HT-free or HT-infected.

Training the Proposed DetectANN. The proposed DetectANN is trained offline. To build the training set, applications are first executed in an HT-free system while runtime attributes are recorded. These attributes are used for the input layer of DetectANN, and the desired output is HT-free. Then, the same applications are executed multiple times with HT-infected NoC components. For a better training result, HTs are randomly implanted each time. DetectANN monitors the same attributes, and routers with implanted HTs are labeled as HT-infected.

Avoidance of False-Positives/Negatives. As discussed, inaccurate HT-detection can lead to performance degradation. False-positives can be a problem when an HT-free router is always labeled as HT-infected. In TSA-NoC, even if the HT-free router is mistakenly labeled as HT-infected, the detection result will be updated at the next epoch. As the trained DetectANN has a high HT-detection accuracy, the wrongly labeled router has a high chance to be labeled correctly at the next epoch. By doing so, the penalty of isolating that HT-free router will be limited to one epoch. Therefore, the false-positive problem can be mitigated. False-negatives are common in conventional designs, in which an HT-infected router is labeled as HT-free when the HT is not activated. The proposed DetectANN resolves this problem by monitoring the runtime NoC behaviors consecutively and providing HT-detection results every 2000 cycles. As the DetectANN utilizes the average attribute values within the epoch, it is able to sensitively capture the anomaly behavior of HT-infected routers, even if the HTs are triggered on for a short period of time. Therefore, the false-negative problem is resolved.

Learning-Based Dynamic HT-Mitigation Using SmartRoute

We propose a learning-based HT-mitigation mechanism for efficient HT-isolation. We implement a bypass channel and a per-router SmartRoute controller to dynamically route high-security packets without traversing HT-infected

components and utilize the bypassed routers to propagate low-security packets without degrading network performance. There is no need to restrict the transmission paths of the low-security packets, since they are already HT-infected.

When isolating HT-infected routers with bypass channels, the simple switch logic of the bypass channel could limit the throughput of given path directions. TSA-NoC addresses this problem by intelligently balancing traffic-loads with various routing algorithms (O1TURN, West-First, and Negative-First) using a SmartRoute controller. The rationale behind this design is two-fold: 1) avoid injecting into bypass channels and 2) optimize the worst case throughput of different NoC traffic patterns.[10] The O1TURN routing dynamically applies XY or YX routing for each packet to better utilize the network spatially under normal traffic loads. West-First and Negative-First restrict different types of turns and achieve lower latency and less dynamic power consumption than O1TURN under intense traffic loads.[10] Note that the TSA-NoC router has multiple virtual channels to avoid protocol and routing deadlocks.

Since the HT-detection results from DetectANN vary periodically during runtime, selecting the most suitable routing algorithm that can handle the dynamic interactions between diverse traffic patterns and limited NoC resources is complex. Therefore, we propose the use of DRL to automatically balance the tradeoffs among the different routing algorithms to achieve better system-level performance for high-security and low-security packets.

DRL-Based Control Policy. The adaptive routing algorithm is applied iteratively to avoid the timing overhead incurred by NoC reconfiguration and packet draining. The length of each iteration (epoch) is identical to that of DetectANN. At each epoch, the DRL-based SmartRoute controller monitors NoC attributes and suggests an action (applying one of the routing algorithms) with the highest expected long-term return[11] in terms of network performance and energy efficiency. The network attributes will change with the action selection, resulting in a new state at the next epoch. The changes in performance and energy metrics are evaluated to update the reward accordingly. The DRL-based control policy continues to evolve based on the NoC historical activities and generates a direct map between the optimal action and a given state. The problem formulation is as follows.

State and Action Space. We select a set of network-related attributes to represent the state vector s for SmartRoute, which include the 12 attributes used in DetectANN, local router label (HT-free or HT-infected), and packet injection rates of different network dimensions. The action space $\{a_1, a_2, a_3\}$ comprises three routing algorithms: O1TURN, West-First, and Negative-First.

Reward Function. The goal of the DRL agent is to select actions that can maximize the long-term return $\mathcal{R}$ for any given state. In SmartRoute, we use Q-learning[11] to estimate the expected long-term return for each state-action pair, recorded as $\mathcal{R} = Q(s, a)$. At each epoch, the agent selects the action with the highest $Q(s, a)$. Next, it observes the immediate reward r and the new state s'. The $Q(s, a)$ value is updated using the following rule:

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s,' a')]. \quad (1)$$

The variables α and γ are DRL parameters called learning rate and discount rate, respectively. The immediate reward r implies minimizing the latency and power consumption. Therefore, we define the immediate reward r_t in (1) at epoch t as follows:

$$r_t = (Power_t \cdot Latency_t)^{-1} \quad (2)$$

The $Latency_t$ and $Power_t$ values are obtained by average end-to-end latency and power consumption (static and dynamic), respectively. The DRL agents select actions according to the Q-table. To eliminate storing overheads, the Q-table in conventional RL is replaced with a neural network.

The working of the DRL of SmartRoute controller is shown in Figure 3. At each epoch, the router first uses the feature values in the state vector s as inputs of the expanded ANN. The ANN then calculates the Q-values of all the possible state-action pairs in the state entry. The router suggests an action a, which has the maximum $Q(s, a)$-value for the next epoch. All routers vote with their selected actions for packets that require HT-free transmission paths, and the routing algorithm with the highest score is selected.

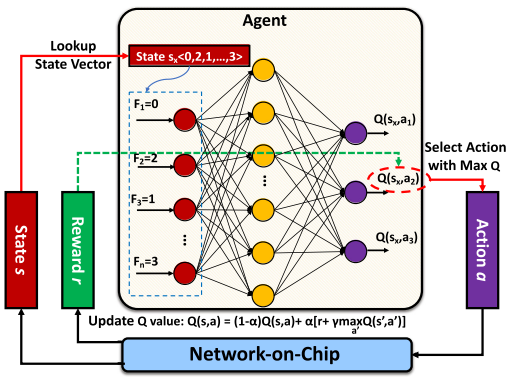

Figure 3. Working of DRL of SmartRoute.

Upon taking the action a, the NoC system transitions to a new state s'. The NoC system then provides an immediate reward r, which is used to update $Q(s, a)$. We implement a five-cycle window between two consecutive epochs to inform routers of the upcoming actions and store on-the-fly flits in router buffers.

EVALUATION AND ANALYSIS

Simulation Setup

We implement the proposed TSA-NoC architecture in the Gem5 full-system simulator. We implement 64 out-of-order CPUs with 2-level cache in an 8×8 2D-mesh network. Additionally, we implement a runtime error injection module consisting of NoC fault and thermal models (DSENT, HotSpot, and VARIUS) to realistically simulate transient errors. We compare the performance of TSA-NoC (DetectANN+SmartRoute) with three techniques, namely FHL+SurfNoC, DetectANN+SurfNoC, and DetectANN+NIBR, with the PARSEC benchmark. We train the DetectANN and SmartRoute with a semi-real dataset generated with synthetic traffic and part of PARSEC benchmark applications (dedup, facesim, freqmine, and swaption). The rest of real applications in PARSEC benchmark are used in the testing phase.

Before executing each benchmark application, we randomly select 10% of the total transmission links in the NoC and implant HTs in them for runtime fault injection. The testing phase of each application lasts for the entire application execution time. The area overhead of TSA-NoC is evaluated using the Synopsys design compiler with the 32-nm library.

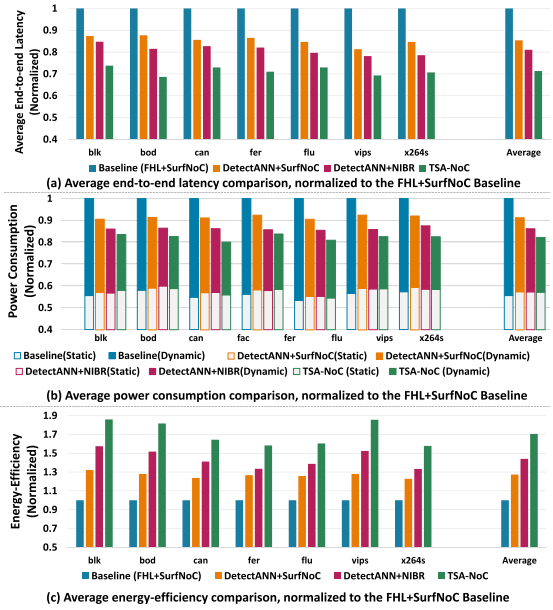

Figure 4. Simulation results of the proposed TSA-NoC: (a) average end-to-end latency, (b) average power consumption, and (c) average energy efficiency. Results are normalized to the FHL+SurfNoC baseline.

For DetectANN and SmartRoute, we set the epoch size to 2000 cycles. The Q-values are initialized to 0. The learning rate α and the discount rate γ are set to 0.1 and 0.9, respectively. Additionally, the DRL agents have a small probability of ϵ = 0.05 to select a random action for exploring unvisited state–action pairs.

Performance Analysis

Average Network Latency: Figure 4(a) shows the normalized average end-to-end packet latency of all the transmitted packets. TSA-NoC achieves an average of 29% end-to-end latency reduction over the baseline. Note that the proposed TSA-NoC using SmartRoute can improve upon DetectANN+SurfNoC by an additional 13% over baseline. This illustrates that DRL-based dynamic routing can further improve overall network latency.

Power Consumption: We evaluate static and dynamic power consumption for all the techniques used. For TSA-NoC, the power consumption of the learning-based TSA-router (with DetectANN and SmartRoute), intermediate links, and bypass channels are included. We first model the static power of all components with Synopsys Design Compiler. Afterward, the captured power

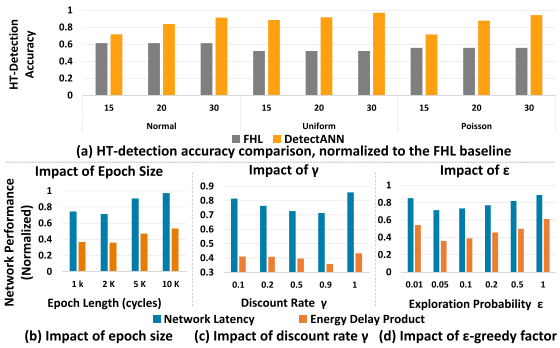

Figure 5. Sensitivity study: (a) HT-detection accuracy, (b) epoch size, (c) discount rate γ, and (d) ϵ-greedy factor.

parameters are fed to the full-system simulator for accurate dynamic power simulation. Figure 4(b) shows that TSA-NoC reduces overall power consumption by an average of 18% over the baseline. The majority of power saving is from dynamic power reduction.

Energy Efficiency: Energy efficiency is defined as: packets$\times$energy^{-1}, where energy equals overall power consumption (of all NoC components, the proposed DetectANN, and SmartRoute) multiplies benchmark execution time. Figure 4(c) shows that the proposed framework improves energy efficiency by an average of 70% compared to baseline.

HT-Detection Accuracy: The HT-detection accuracy is calculated with the ratio of the number of identified HTs to the total number of implanted HTs within a full execution of each benchmark. In this simulation, we vary the middle layer size of DetectANN. The DetectANN is trained with random distributed HT-generated bit-flips, while in the testing phase, the HT-generated bit-flips follow three different distributions: normal, uniform, andPoisson distribution. Figure 5(a) shows that, for all distributions, the proposed DetectANN improves HT-detection accuracy by 39% on average over the FHL baseline, with 30 neurons in the middle layer.

Sensitivity Analysis

Impact of Middle Layer Size of DetectANN: We vary the size of the middle layer to study its impact on HT-detection accuracy in Figure 5(a). The HT-detection accuracy improves as the size of the middle layer increases. For the best accuracy, area consumption, and timing overhead, we use 30 neurons in the middle layer.

Impact of Epoch Size of DRL: In this test, we vary the length of DRL epoch from 1000 to 10 000 clock cycles. As shown in Figure 5(b), increasing the epoch size negatively impacts network latency and energy-delay product (EDP) due to coarse-grain control (lower network latency and EDP indicate better performance). Alternatively, aggressively reducing the length of epochs also leads to performance degradation, as the timing overhead of DRL would be notable.

Impact of Discount Rate γ of DRL: Figure 5(c) depicts the impact of the discount rate γ on network performance. As shown in Figure 5(c), both network latency and EDP are initially improved with larger γ. However, aggressively increasing γ can also result in slow DRL convergence, which leads to performance degradation. The best performance is achieved when γ equals 0.9.

Impact of Exploration Factor ϵ of DRL: The impact of ϵ on network performance is shown in Figure 5(d). As ϵ increases, the agent explores unvisited state-action pairs more frequently, which is beneficial for training DRL. However, when ϵ equals 1, the router will take actions completely at random. As shown in Figure 5(d), the best performance is achieved when ϵ equals 0.05.

Overhead Analysis

Timing Overhead: The timing overhead is induced by calculating and updating the weights for DetectANN and SmartRoute. In the worst case, for each epoch, the computation overheads are 90 and 150 cycles for DetectANN and SmartRoute, respectively. We use two sets of different epochs for the monitoring and controlling to minimize the negative effect of this latency. The two sets of epochs are offset by the ANN computation time, which can pipeline the overhead effectively. By doing so, the calculation of ANNs does not block monitoring or controlling. Therefore, the use of ANN does not negatively impact the overall performance.

Area and Power Overhead: The proposed DetectANN and Smart-Route require additional ALUs (adders, multipliers, and Sigmoid function) and SRAM storage in each router. The proposed DetectANN consumes additional 425.2-μm^2 area for ALUs and 718.7-μm^2 area for SRAM, incurring 0.9% area overhead over a conventional router

in total. The DRL logics consume 956.7-μm^2 ALU area and 1617.2-μm^2 SRAM area, which implies 2.1% area overhead. Furthermore, the power overheads of DetectANN and DRL are 0.086 and 0.195 MW, respectively.

CONCLUSIONS

In this article, we proposed TSA-NoC, a learning-enabled, high-performance, and energy-efficient design framework for secure on-chip communication. The TSA-NoC consists of an ANN-based HT-detection design (DetectANN) and a DRL-based adaptive routing mechanism (SmartRoute). The proposed DetectANN detects HT-infected NoC components promptly and accurately at runtime. SmartRoute isolates the HT-infected components and deploys dynamic routing algorithms to optimize system-level performance. Full-system evaluations show that TSA-NoC improves HT-detection accuracy, network latency, and energy efficiency over existing techniques.

ACKNOWLEDGMENTS

This work was supported by NSF under Grants CCF-1547035 and CCF-1702980.

REFERENCES

1. H. Wassel *et al.*, "Networks on chip with provable security properties," *IEEE Micro*, vol. 34, no. 3, pp. 57–68, May/Jun. 2014.
2. T. Boraten and A. K. Kodi, "Securing NoCs against timing attacks with non-interference based adaptive routing," in *Proc. 12th IEEE/ACM Int. Symp. Network-on-Chip*, 2018, pp. 1–8.
3. H. Wassel *et al.*, "SurfNoC: A low latency and provably non-interfering approach to secure networks-on-chip," in *Proc. 40th Annu. Int. Symp. Comput. Archit.*, 2013, pp. 583–594.
4. M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Des. Test Comput.*, vol. 27, no. 1, pp. 10–25, Jan./Feb. 2010.
5. D. M. Ancajas, K. Chakraborty, and S. Roy, "Fort-NoCs: Mitigating the threat of a compromised NoC," in *Proc. 51st ACM/EDAC/IEEE Des. Autom. Conf.*, 2014, pp. 1–6.
6. K. Xiao and M. Tehranipoor, "BISA: Built-in self-authentication for preventing hardware Trojan insertion," in *Proc. IEEE Int. Symp. Hardware-Oriented Secur. Trust*, 2013, pp. 45–50.
7. H. Salmani, "COTD: Reference-free hardware trojan detection and recovery based on controllability and observability in gate-level netlist," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 2, pp. 338–350, Feb. 2017.
8. K. Wang *et al.*, "IntelliNoC: A holistic design framework for energy-efficient and reliable on-chip communication for manycores," in *Proc. 46th Annu. Int. Symp. Comput. Archit.*, 2019, pp. 1–12.
9. Q. Yu and J. Frey, "Exploiting error control approaches for hardware Trojans on network-on-chip links," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst.*, 2013, pp. 266–271.
10. A. Singh. "Load-balanced routing in interconnection networks," Ph.D. thesis, Stanford Univ., Stanford, CA, USA, 2005.
11. R. Sutton *et al.*, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

Ke Wang is currently working toward the Ph.D. degree in computer engineering with George Washington University. His research work focuses on high-performance, energy-efficient, and reliable interconnect designs using machine learning-based optimization. Contact him at cory@gwu.edu.

Hao Zheng is currently working toward the Ph.D. degree in computer engineering with George Washington University. His research interests are in the areas of computer architecture and parallel computing, with emphasis on on-chip interconnects. Contact him at haozheng@gwu.edu.

Ahmed Louri is currently the David and Marilyn Karlgaard Endowed Chair Professor of Electrical and Computer Engineering with George Washington University. Louri received the Ph.D. degree in computer engineering from the University of Southern California in 1988. He conducts research in the broad area of computer architecture and parallel computing. He is a Fellow of IEEE, and currently serves as the Editor-in-Chief of the IEEE TRANSACTIONS ON COMPUTERS. Contact him at louri@gwu.edu.