AGAPE: Anomaly Detection with Generative Adversarial Network for Improved Performance, Energy, and Security in Manycore Systems

Ke Wang*, Hao Zheng†, Yuan Li*, Jiajun Li*, Ahmed Louri*
*Dept. of Electrical and Computer Engineering, George Washington University, Washington, DC, USA
†Dept. of Electrical and Computer Engineering, University of Central Florida, Orlando, Florida, USA
Emails: {cory, haozheng, liyuan5859, lijiajun, louri}@gwu.edu

***Abstract*—The security of manycore systems has become increasingly critical. In system-on-chips (SoCs), Hardware Trojans (HTs) manipulate the functionalities of the routing components to saturate the on-chip network, degrade performance, and result in the leakage of sensitive data. Existing HT detection techniques, including runtime monitoring and state-of-the-art learning-based methods, are unable to timely and accurately identify the implanted HTs, due to the increasingly dynamic and complex nature of on-chip communication behaviors. We propose AGAPE, a novel Generative Adversarial Network (GAN)-based anomaly detection and mitigation method against HTs for secured on-chip communication. AGAPE learns the distribution of the multivariate time series of a number of NoC attributes captured by on-chip sensors under both HT-free and HT-infected working conditions. The proposed GAN can learn the potential latent interactions among different runtime attributes concurrently, accurately distinguish abnormal attacked situations from normal SoC behaviors, and identify the type and location of the implanted HTs. Using the detection results, we apply the most suitable protection techniques to each type of detected HTs instead of simply isolating the entire HT-infected router, with the aim to mitigate security threats as well as reducing performance loss. Simulation results show that AGAPE enhances the HT detection accuracy by 19%, reduces network latency and power consumption by 39% and 30%, respectively, as compared to state-of-the-art security designs.**

***Index Terms*—Manycore systems, Security, Hardware Trojans (HTs), Generative Adversarial Networks (GAN)**

I. Introduction

As technology scales, System-on-Chip (SoC) designs have evolved to complex manycore systems. Network-on-Chip (NoC) [1], as the standard interconnection fabric for SoCs, has become the security vulnerability of manycore systems, since it has visibility of the entire SoC and access of communications among cores [2]–[9]. In NoC, Maliciously implanted Hardware Trojans (HTs) [2], [5], [9], including fault-injection HTs, misrouting HTs, packet-drop HTs, and more, have been shown to destruct NoC functionality, degrade NoC performance, leak information, and covertly transmit data.

Previous work has been devoted to secure NoCs by detecting and mitigating HT attacks [3], [5], [8], [10], [11]. Conventional HT detection techniques use runtime monitoring [4], [6], [12]–[15] to detect HT-infected components by capturing abnormal attribute values (e.g., injection rate, buffer/link utilization, and latency) that far exceed a manually designed threshold. However, since communication and network behaviors are hard to be predicted or modeled in the complex manycore system, it is impossible to manually select the attribute thresholds without resulting in inaccurate detection and performance penalties.

State-of-the-art methodologies [16]–[19] use machine learning algorithms to eschew human-designed thresholds for HT detection. However, these techniques require a significant amount of pre-labeled data samples to train, which requires excessive human engineering. Besides, in those designs, although the training data samples are generated with real applications, these captured data sets can hardly cover sufficient design space of possible HT behaviors (e.g. fault/error injections) to engender confidence in the training accuracy. Furthermore, these techniques usually employ the attribute vector of instantaneous attribute values for detection. This can be inadequate for anomaly detection since the impact of HTs on NoC behaviors is often reflected by the hidden inherent correlations of multivariate time series of these attributes. For HT mitigation, existing designs [3], [8] isolate the HT-infected routers and deploy adaptive/regional routing to reallocate the traffic, relieve network congestion, and provide non-interference transmissions. However, these techniques rely on rerouting packets which may not be efficient for all types of HTs and often cause significant increases in network latency and power consumption [3].

To address these challenges, we propose AGAPE, which is a novel NoC security framework for HT detection and mitigation using a semi-supervised generative adversarial network (GAN). The proposed GAN-based HT detection design models the correlations among multiple time series of a number of runtime NoC attributes and detect anomalies based on the trained GAN model. The GAN model [20] is comprised of a generator and a discriminator, both of which are constructed with neural networks and trained iteratively. The generator generates sufficient realistic training samples to significantly improve the coverage of possible cases in the training phase, followed by feeding the generated training sets to the discriminator which tries to distinguish their correct labels. Once trained, the generator can capture the hidden distributions and interactions of the training sequences, and the discriminator is able to distinguish HT-infected scenarios from normal NoC behaviors with high sensitivity. Using the HT detection results provided by the proposed GAN, we apply the most suitable protection techniques to each type of detected HTs instead of simply isolating the entire HT-infected router, with the aim to mitigate security

threats as well as reducing performance loss. Specifically, we enhance the router architecture with dynamic error correction hardware to correct the faults induced by the fault-injection HTs, and bypass links and channel buffers are used to avoid the malfunctioning router components infected by misrouting HTs and packet-drop HTs, respectively. Experimental studies using full-system simulation indicate that AGAPE enhances the HT detection accuracy by 19%, as compared to state-of-the-art learning-based HT detection method, and achieves up to 91% HT detection accuracy. Furthermore, by applying suitable protection mechanisms, the proposed framework reduces network latency and power consumption by 39% and 30%, respectively, as compared to existing NoC security designs.

II. Background and Related Works

A. Hardware Trojans in On-chip Networks

Hardware Trojans (HTs) [2], [4]–[6], [9], [13], [16], [21] are intentional hardware alterations of the design specification or of the corresponding implementation. HTs are implanted during the IC design phase of the circuits. After being implanted, the HTs usually remain dormant to avoid being detected and are activated upon internal or external triggering events. In NoCs, HTs [2], [5], [9], [13] are implanted in diverse hardware components of the routers. Fault-injection HTs are implanted at output ports or the error checking hardware (ECC) of the routers. The implanted fault-injection HTs can force data bit-flips of the transmitted packet and incur retransmission traffic. The misrouting HTs are implanted in the pipeline stages (e.g. routing computation logic). Once activated, they force the packet to be transmitted via the routes that are forbidden by the routing algorithm to cause dead-lock and live-lock problems. Another type of HTs, the packet-drop HTs, are implanted in the input buffers which can discard all the incoming flits stored in the infected input virtual channel. These HTs can downgrade performance by intentionally incurring retransmission traffic, extensively increasing end-to-end latency, and creating network congestion which can significantly disrupt traffic.

In this paper, we focus on the attack model in which fault-injection HTs, misrouting HTs, and packet-drop HTs, are randomly implanted into some of the routers in the NoC. The triggering event for fault-injection HTs is runtime temperature (router chip temperature) [22], while misrouting HTs and packet-drop HTs are triggered by local buffer utilization [7].

B. Existing HT Detection Methodologies

Existing works have explored both traditional runtime monitoring methods and machine-learning-based designs for HT detection. The former HT detection techniques [4], [6] monitor NoC attributes and detect malicious components by capturing abnormal attribute values (e.g., injection rate, buffer/link utilization, and latency) that far exceed a manually designed threshold. The threshold values, if not carefully selected, can result in inaccurate detection, thus leading to performance degradation and security vulnerabilities. For example, the false-positive detection results, when an HT-free router is marked as HT-infected, can lead to isolating the HT-free router and packet re-routing, therefore increasing the network latency. The false-negative results leave the HT-implanted routers untreated, which can result in an insecure system. Machine-learning-based designs use supervised learning and neural networks [16]–[19] to improve the HT detection accuracy. However, these techniques, when being used in complex systems like NoCs, require substantial training data, which includes millions of normal data and labeled anomaly data recorded in time series of all the related runtime NoC attributes. Moreover, these existing methodologies are always specifically designed and trained for one type of HT, thus are not able to capture the hidden inherent correlations of all attributes in a complex system, where multiple HT models are implanted simultaneously.

III. Hardware Trojan Detection with Generative Adversarial Network

We propose AGAPE, a learning-based NoC design framework that provides high-performance and power-efficient detection and mitigation against HTs. In this section, we introduce AGAPE's HT detection methodology using an improved semi-supervised GAN [23].

A. GAN Basics

The Generative Adversarial Networks (GANs) [23] have been widely used for anomaly detection [20], [24], [25]. The generative model of GANs follows an adversarial training process, in which two neural networks, a discriminator D and a generator G, are simultaneously trained. In GAN, both G and D are multi-layered conventional neural networks. The generator G takes as input a random noise vector z and outputs synthesized fake training samples (e.g. time series of a set of NoC attributes). The discriminator D receives a real training sample x or a synthesized sample generated by G and identifies the source of the received training sample. Designed based on the game-theoretic formulation, the discriminator and generator are considered to be two players which compete against each other in a zero-sum min-max game, followed by a value function $V(G, D)$:

$$\min_G \max_D V(D,G) = \mathbb{E}_x[logD(x)] + \mathbb{E}_z[log(1 - D(G(z)))] \quad (1)$$

Trained in the opposition to each other, the generator G can eventually generate synthesized samples that are extremely similar to the real samples. In this case, the generator G is able to learn and capture the potential latent interactions among the runtime attributes concurrently. The discriminator D, trained with both real samples and synthesized samples, would be extremely sensitive to distinguish the correct label of the testing samples and can accurately capture the anomaly in the training data sequences for anomaly detection.

B. Proposed HT Detection Framework

The goal of the proposed AGAPE design is to not only detect anomalies in runtime NoC attributes when HTs are implanted, but also to correctly identify the type of the implanted HTs. Therefore, we extend the conventional GAN model with class labels c for both G and D. The class labels $c \in \{c_n, c_f, c_m, c_p\}$ represents the scenarios in which no HTs are implanted, fault-injection HTs are implanted, misrouting HTs are implanted,

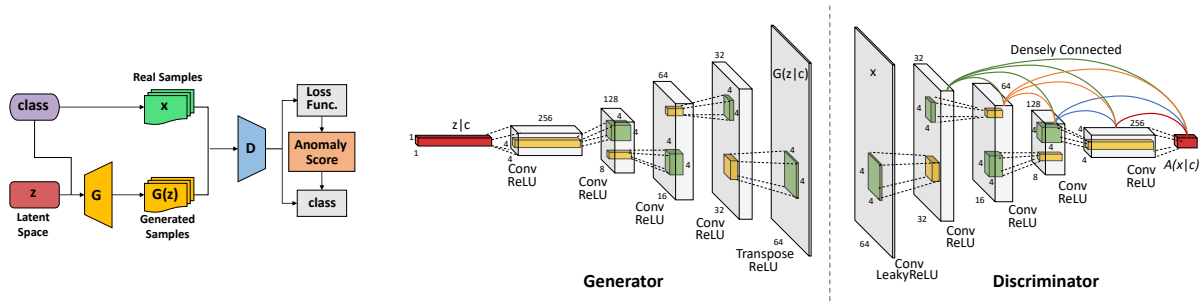

Fig. 1. The architecture of the proposed GAN. (a) shows the overall architecture and the workflow. (b) shows the design of the generator (G) and GAN discriminator (D). The generator and discriminator are both constructed with convolutional layers. The generator captures the latent space z and generates samples $G(z)$. The discriminator uses the feature map captured by the generator and real samples x to calculate the anomaly detection result $\mathcal{A}(x|c_i)$.

and packet-drop HTs are implanted, respectively. The proposed GAN model is shown in Fig. 1. (a). The generator G uses the class label and the training sample as inputs to generate the synthesized time series. The discriminator takes a testing sample as input and calculates the results if the input sample x is from G or real training sample space as well as the classification of the training sample.

Fig. 1.(b) shows the designs of the generator and the discriminator of the proposed GAN. First, time series of the runtime attributes used for the input layer of the generator are split into small sequences and reconstructed into matrices. Then the input is fed to the encoder part of the generator, which has four convolutional layers to capture the feature maps. The kernel size is 4 * 4. After that, a transpose layer is used as the decoder part of the generator to generate the synthetic training set from the feature maps. The output layer is of the same size as the input matrix. The activation function used for all generator layers is ReLU. The discriminator uses the feature map matrices as input. The discriminator has five convolutional layers with 20% dropout, and the filter size is 4 * 4. LeakyReLU is used for the activation function between the first two convolutional layers. The last four convolutional layers in the discriminator are densely connected where the activation function is ReLU. Specifically, to keep the feed-forward quality, each of the last four layers in the network of the discriminator obtains additional input from all the previous layers. The output layer has four neurons. Our GAN was trained offline. To reduce the area overhead of GAN implementation, only the trained discriminator is embedded in the proposed router.

The training and the testing samples are comprised of sets of time series of run time NoC attributes. These NoC attributes include 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), packet injection rate of each input port, retransmission rate (NACK rate) of each output port, average end-to-end latency of the propagated packets, and local operation temperature. These input values are normalized in the range of 0 to 1 for the non-linearity of the activation functions. These attributes are measured every 100 cycles and recorded in the time series for GAN training for each 5000 cycle iteration. The generator and discriminator are trained following the minimax game within each class c:

$$\min_G \max_D V(D,G|c) = \mathbb{E}_{x,c}[logD(x|c)] + \mathbb{E}_{z,c}[log(1-D(G(z|c)))] \quad (2)$$

The real samples x are captured with the NoC system running synthetic traffic and real applications. These samples are recorded and labeled in four classes, $c \in \{c_n, c_f, c_m, c_p\}$. The designed AGAPE is trained multiple rounds with each class of real samples respectively.

Once trained, the generator G is able to learn the potential interactions among the NoC attributes and capture the mapping from latent space z to real samples x. Using this learned mapping $G(z) = z \mapsto x$, during the inference phase, the discriminator takes the testing sample, or the time series, and maps the sample to the latent space. The discriminator D then calculates the output of the discriminator CNN and the loss function $\mathcal{L}$. The value of $\mathcal{L}$ indicates the similarity of the testing sample to the training samples, which can be used to generate an *anomaly score* $\mathcal{A}$ for HT detection. Specifically, the loss function $\mathcal{L}$ if comprised of a residual loss $\mathcal{L}_R$ and a discrimination loss $\mathcal{L}_D$, which are defined as:

$$\mathcal{L}_R = \sum |x - G(z_\gamma)| \qquad \mathcal{L}_D = \sum |f(x) - f(G(z_\gamma))| \quad (3)$$

where function f is the sigmoid cross-entropy. The residual loss $\mathcal{L}_R$ measures the dissimilarity between the testing sample and the training sample $G(z_\gamma)$ in the latent space. For an ideal normal testing sample, the value of $\mathcal{L}_R$ is zero. The discrimination loss $\mathcal{L}_D$ indicates how well the training sample is mapped to the latent space. We define the anomaly score $\mathcal{A}$ within class c as: $\mathcal{A}(x|c) = (1-\lambda) \cdot \mathcal{L}_R(x|c) + \lambda \cdot \mathcal{L}_D(x|c)$, where λ is set to 0.5, and samples with $\mathcal{A}$ below 0.9 is classified as anomalies. In the proposed AGAPE design, each testing sample will be used to calculate four anomaly scores for all classes. The class with the smallest $\mathcal{A}(x|c_i)$ under 0.9 would be the detection result. In doing so, during the inference phase, the discriminator can justify whether an anomaly exists in the testing phase and also identify the type of the implanted HT.

IV. Threat Mitigation Techniques

We propose a set of efficient HT mitigation techniques for fault-injection, misrouting, and packet-drop HTs, respectively.

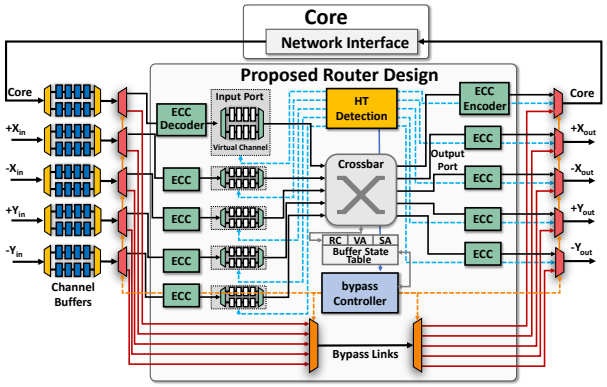

Fig. 2. The proposed AGAPE router architecture for HT mitigation.

The proposed AGAPE router architecture is shown in Fig. 2. We enhance the conventional router architecture with three dynamic hardware designs, namely an HT detection module using the proposed GAN architecture, a set of pass links for infected router isolation, and a set of channel buffers in each port to improve the throughput and latency when HT infected router components are isolated. These proposed designs can dynamically be enabled or disabled according to the HT results.

For fault-injection HTs, a double-error correction, triple-error detection code (DECTED) is applied to the HT-infected router, which is capable of correcting 2 error bits in each transmitted flit automatically. Once the fault-injection HT is detected, the router enables the DECTED encoder located at the output ports and sends an activation signal to its adjacent downstream routers to enable their DECTED decoders. During packet transmission, each flit is first encoded with the error correction code, and it will be verified with the decoder of the downstream router. Meanwhile, a copy of the transmitted flit is buffered in the router's virtual channel until it receives an ACK message back from the downstream router. If the flit fails the error checking, a negative-ACK will be sent back to the upstream router, and the buffered flit will be retransmitted. This can mitigate the HT-injected faults in the transmitted data without fully isolating the HT-infected router.

For misrouting HTs and packet-drop HTs, we modify the NoC architecture with router bypass links and channel buffers to transmit packets without traversing the HT-infected routing computation logic and input buffers. Specifically, the bypass links connect all the input and output ports of an isolated router with the bypass links and a simple switch logic using MUXes/DEMUXes. The packets being transmitted are propagated via the bypass links using a round-robin scheme. To fully bypass the routing computation logic, the routing information is recorded in a modified buffer state table. Similar to the VC state table, the modified buffer state table consists of the following entries: the VC identifier (VC), read pointer (RP), write pointer (WP), allocated output ports (OPX and OPY), output VC (OVC), state (ST), and credit count (CR). Additionally, channel-buffer-related entries are added, namely input port identifier (Port) which identifies the input port of the incoming flit, downstream router status (DRS) which indicates whether the downstream router is available or bypassed/power-gated, a channel buffer pointer (CBP), and the channel buffer credit (CBC) which gives the occupancy status of the associated channel buffers. The header flit carries the packet information for route computation (RC) and VC allocation. If the router input buffer is HT-infected and needs to be bypassed, the buffer allocation table assigns a free channel buffer slot to the header flit and records the VC information (VC) and output information (OVC, OPX, and OPY). Thus, the body flits can be routed to the associated output port by simply following the VC to find the correct output port from the buffer allocation table. The credit (CR) is updated in the buffer allocation table. This guarantees that the flow control can operate normally irrespective of whether the router is available or bypassed. Additionally, all the available router buffer slots and channel buffer slots are recorded in a channel buffer state table. If all the router buffer slots and channel buffer slots of an input direction are occupied, a congestion signal will be generated.

V. Evaluation and Analysis

A. Simulation Setup

We evaluate the proposed AGAPE design using the GEM5 full-system simulator, in which we fully incorporate the HT attack models, the proposed AGAPE, and the HT mitigation techniques. The generator and discriminator are trained offline before being implemented in each router for the testing phase. In the testing phase, different types of HTs are randomly implanted. At runtime, these HT-infected routers are activated when the triggering events are met. Workloads from the PARSEC benchmark suite are tested.

We compare the performance of the proposed design to three other solutions, namely (1) baseline: a baseline which consists of a traditional wormhole-based router with conventional runtime monitoring [4] for HT detection and a basic router-isolation strategy for detected HTs, (2) RM: conventional runtime monitoring and the proposed HT mitigation technique, and (3) ANN: state-of-the-art artificial neural network (ANN)-based HT detection [17], [18] and the proposed HT mitigation technique. During the testing phase, each benchmark application is executed for ten rounds, HTs are randomly implanted for each round of execution. Performance metrics are recorded using the average measurement values of all testing rounds.

We evaluate the area and power consumption with Synopsys. The value of power consumption refers to the summary of NoC static power and dynamic power. We first modeled the static power of all components with Synopsys. Since Synopsys cannot evaluate the dynamic power accurately for different benchmark applications, we fed the power parameters captured by Synopsys to the DSENT power model which is integrated with GEM5. During the application execution, DSENT calculates the average dynamic power by the number of buffer-writes, crossbar, VA/SA activities, and ANN/DQL calculations within the full application execution time.

B. HT Detection Accuracy

We use precision, recall (or sensitivity), and F1-score, along with the conventional metrics (HT detection rate and detection

TABLE I
HT DETECTION METRICS USING DIFFERENT TECHNIQUES

Technique	HTs	Precision	Recall	F1_score	Detection Rate	Correctness	Accuracy
AGAPE	Fault-injection	**0.8728**	0.9642	**0.9162**	**0.9571**	**0.9512**	**0.9105**
	Misrouting	0.7698	0.9281	0.8416			
	Packet-drop	0.8020	**0.9792**	0.8818			
ANN-based Detection [18]	Fault-injection	0.5227	0.9494	0.6742	0.8324	0.8715	0.7254
	Misrouting	0.4950	0.7588	0.5991			
	Packet-drop	0.4526	0.7890	0.5752			
Runtime Monitoring [4]	Fault-injection	0.1646	0.6742	0.2646	0.6498	0.7459	0.4847
	Misrouting	0.1290	0.6480	0.2152			
	Packet-drop	0.1065	0.6272	0.1821			

accuracy), to evaluate the performance of the proposed AGAPE. The definitions of these measurements are:

$$\begin{aligned} Precision &= \frac{TP}{TP+FP+ITP} \\ Recall(Sensitivity) &= \frac{TP+ITP}{TP+ITP+FN} \\ F1\_score &= 2 \cdot \frac{Recall \cdot Precision}{Recall + Precision} \\ DetectionRate &= \frac{\Sigma(TP+ITP)}{Number\ of\ implanted\ HTs} \\ Correctness &= \frac{TP}{TP+ITP} \\ Accuracy &= DetectionRate \cdot Correctness \end{aligned} \tag{4}$$

where TP is true positives, FP is false positives, and FN is false negatives. It should be noted that the HT detection techniques may categorize some HTs as other types. In such cases, we consider the detection result as incorrect true positives (ITP). We select these metrics to indicate the potential impact of the detection results on overall system-level NoC performance. For example, a higher precision value means fewer false positives, where HT-free routers are marked as HT-infected and isolated falsely. This will result in reduced network latency penalties. A higher recall value implies fewer false negatives (the undetected HTs which are security threats). The F1_score, which is a weighted average of precision and recall, takes both false positives and false negatives into account and is often considered to be a better measurement of the capability to identify real threats when classes are uneven distributed (e.g. more actual TNs). The correctness value indicates how well the detection techniques can correctly label the detected HTs.

The entire PARSEC benchmark is tested, and the average measurement values are listed in Table. I. As shown in Table. I, across all types of HTs, the proposed AGAPE achieves over 0.84 F1_scores, indicating improvements in both precision and recall. The proposed AGAPE improves overall detection accuracy by 18.5% over state-of-the-art ANN-based HT detection and achieves over 95% overall detection rate, 95% correctness, and 91% HT detection accuracy, due to the use of GAN and time series of the NoC attributes.

C. Performance Analysis

Speedup: The speedup is calculated with the full execution time of each benchmark application, using various HT detection techniques (baseline, RM, ANN, and AGAPE), as shown in Fig. 3. As can be seen in Fig. 3, simply deploying dynamic HT mitigation technique has limited speedup because of the inaccurate HT detection. The proposed AGAPE design achieves an average of 16%, 14%, and 9% speedup over the other three designs, respectively, thanks to the improved HT detection accuracy, which can eliminate the performance loss induced by applying incorrect HT mitigation techniques.

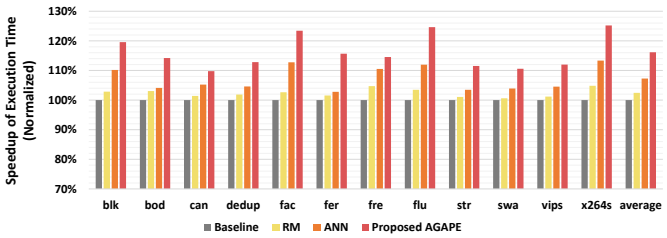

Fig. 3. Speedup of full application execution time comparison, normalized to the baseline (higher is better).

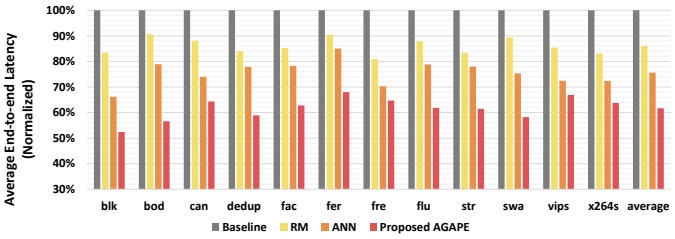

Fig. 4. Average end-to-end latency comparison, normalized to the baseline (lower is better).

Average End-to-End Latency: Fig. 4 shows the normalized end-to-end packet latency for different techniques. It can be seen in Fig. 4, the end-to-end latency is improved by 14% by solely using the proposed dynamic HT mitigation technique. It is because the proposed dynamic HT mitigation technique eliminates the isolation of the entire HT-infected routers and increases network throughput with bypass links and channel buffers. For the fault-injection HTs, the proposed HT mitigation technique corrects the injected faults in the packets to reduce retransmission packets, which consumes network resources and can lead to network congestion. As compared to RM and ANN, the proposed AGAPE improves end-to-end latency by 25% and 14%, respectively. This indicates that the proposed HT mitigation technique achieves greater network latency reduction with higher HT detection accuracy.

Power Consumption: We evaluate the overall power consumption, which is comprised of static and dynamic power consumption, as shown in Fig. 5. The RM design reduces the overall consumption by 9%, as compared to the baseline, due to the use of the proposed dynamic HT mitigation technique. AGAPE reduces power consumption by additional 21% and 12% over RM and ANN, respectively. That is because the proposed design can reduce static power consumption using router bypassing and mitigates unnecessary packet retransmis-

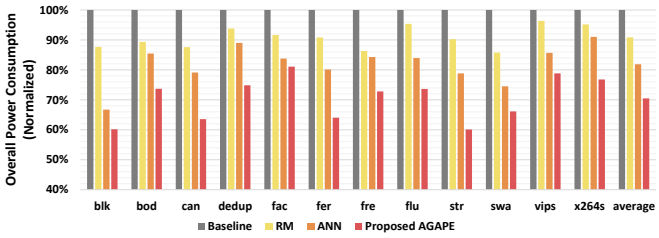

Fig. 5. Overall power consumption comparison, normalized to the baseline (lower is better).

sion and misrouted traffic propagation, thus reducing dynamic power consumption.

D. Overhead Analysis

We evaluate the overheads of AGAPE in terms of timing, chip area, and power. The timing overhead is obtained by GEM5, while the offline-training time is not included. The chip area and power overheads are evaluated using Synopsys Design Vision software with 32nm technology.

The timing overhead is induced by calculating the output of the GAN discriminator during the testing phase. In the worst case, the computation overhead of the discriminator is 270 cycles, which implies 5.4% of the 5000 cycle HT detection interval. To avoid the potential negative impact on network performance, we use two sets of different time steps for attribute monitoring and controlling. The two sets of time steps are offset by the discriminator computation time which can pipeline the overhead effectively. By doing so, the calculation of GANs does not block either the monitoring or the controlling, which will not negatively impact the overall performance.

The proposed GAN-based HT detection requires additional ALUs and SRAM storage in each router for the implementation of the trained discriminator. The proposed design consumes 9161 μm^2 chip area, which implies 7.2% area overhead over a conventional router. However, the proposed AGAPE design only induces 2.9% additional area overhead, as compared to existing ANN-based HT detection designs [18]. Furthermore, the power overhead of the AGAPE is $0.34mw$, which implies 1.5% power overhead over existing ANN-based designs.

VI. Conclusions

We propose AGAPE, a novel Generative Adversarial Networks (GAN)-based anomaly detection method against hardware Trojans (HTs) for high-performance, low-power, and secure on-chip communications in manycore systems. AGAPE learns the distribution and latent interactions among the multivariate time series of a number of NoC attributes under both HT-free and HT-infected working conditions, thus can accurately distinguish abnormal attacked situations from normal NoC behaviors and identify the type and location of the implanted HTs. Using the detection results, the maliciously implanted HTs are treated using suitable HT mitigation techniques efficiently. Specifically, to protect the network against the fault-injection HTs, advanced error correction codes are applied to the HT-infected router to detect and correct the HT-injected faults. The routers with misrouting HTs are bypassed using the bypass links to avoid the faulty routing computation. Furthermore, the packet-drop attacks are mitigated by replacing the infected router input buffers with the channel buffers. Simulation results show that, as compared to current NoC security techniques, the proposed AGAPE framework improves the HT detection accuracy by 19% over state-of-the-art learning-based techniques and achieves up to 91% HT detection accuracy. Moreover, AGAPE reduces network latency and power consumption by 39% and 30%, respectively.

Acknowledgment

This research was partially supported by NSF grants CCF-1702980, CCF-1901165, and CCF-1812495. We sincerely thank the anonymous reviewers for their excellent and constructive feedback.

References

[1] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *DAC'01*, 2001.
[2] J. Rajesh *et al.*, "Hardware trojan attacks in soc and noc," in *The Hardware Trojan War*. Springer, 2018, pp. 55–74.
[3] T. Boraten and A. Kodi, "Securing NoCs against timing attacks with non-interference based adaptive routing," in *NOCS'18*. IEEE, 2018.
[4] H. Salmani, "COTD: Reference-free hardware trojan detection and recovery based on controllability and observability in gate-level netlist," *IEEE TIFS*, vol. 12, no. 2, 2016.
[5] T. Boraten and A. K. Kodi, "Mitigation of denial of service attack with hardware trojans in noc architectures," in *IPDPS'16*. IEEE, 2016.
[6] K. Xiao and M. Tehranipoor, "BISA: Built-in self-authentication for preventing hardware trojan insertion," in *HOST'13*. IEEE, 2013.
[7] R. Karri *et al.*, "Trustworthy hardware: Identifying and classifying hardware trojans," *Computer*, vol. 43, no. 10, pp. 39–46, 2010.
[8] H. M. Wassel *et al.*, "SurfNoC: a low latency and provably non-interfering approach to secure networks-on-chip," in *ISCA'13*, 2013.
[9] P. Mishra and S. Charles, *Network-on-Chip Security and Privacy*. Springer Nature, 2021.
[10] J. A. Ambrose *et al.*, "Rijid: random code injection to mask power analysis based side channel attacks," in *DAC'07*, 2007.
[11] M. Barbosa and D. Page, "On the automatic construction of indistinguishable operations," in *IMACC'05*. Springer, 2005.
[12] A. M. Fiskiran and R. B. Lee, "Runtime execution monitoring (rem) to detect and prevent malicious code execution," in *ICCD'04*. IEEE, 2004.
[13] V. Y. Raparti and S. Pasricha, "Lightweight mitigation of hardware trojan attacks in noc-based manycore computing," in *DAC'19*. IEEE, 2019.
[14] Z. Zhang and Q. Yu, "Invariance checking based trojan detection method for three-dimensional integrated circuits," in *ISCAS'20*. IEEE, 2020.
[15] S. Charles, Y. Lyu, and P. Mishra, "Real-time detection and localization of dos attacks in noc based socs," in *DATE'19*. IEEE, 2019.
[16] T. Iwase, Y. Nozaki *et al.*, "Detection technique for hardware trojans using machine learning in frequency domain," in *GCCE'15*. IEEE, 2015.
[17] K. G. Liakos *et al.*, "Conventional and machine learning approaches as countermeasures against hardware trojan attacks," *Microprocessors and Microsystems*, 2020.
[18] S. Wang *et al.*, "Hardware trojan detection based on elm neural network," in *ICCCI'16*. IEEE, 2016.
[19] Z. Huang *et al.*, "A survey on machine learning against hardware trojan attacks: Recent advances and challenges," *IEEE Access*, vol. 8, 2020.
[20] F. Di Mattia *et al.*, "A survey on gans for anomaly detection," *arXiv preprint arXiv:1906.11632*, 2019.
[21] M. Tehranipoor and K. Farinaz, "A survey of hardware trojan taxonomy and detection," *IEEE design & test*, vol. 27, no. 1, pp. 10–25, 2010.
[22] M. Potkonjak *et al.*, "Hardware trojan horse detection using gate-level characterization," in *DAC'09*, 2009.
[23] I. J. Goodfellow, J. Pouget-Abadie *et al.*, "Generative adversarial networks," *arXiv preprint arXiv:1406.2661*, 2014.
[24] D. Li, D. Chen *et al.*, "Anomaly detection with generative adversarial networks for multivariate time series," 2019.
[25] T. Schlegl *et al.*, "f-anogan: Fast unsupervised anomaly detection with generative adversarial networks," *Medical image analysis*, vol. 54, 2019.