IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 4, 2019, accepted July 13, 2019, date of publication July 16, 2019, date of current version August 9, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2929274

A Benchmark Suite of Hardware Trojans for On-Chip Networks

JIAN WANG, SHIZE GUO, ZHE CHEN, AND TAO ZHANG
University of Electronic Science and Technology of China, Chengdu 611731, China
Corresponding author: Jian Wang (wangjian3630@uestc.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61671110.

ABSTRACT As recently studied, network-on-chip (NoC) suffers growing threats from hardware trojans (HTs), leading to performance degradation or information leakage when it provides communication service in many/multi-core systems. Therefore, defense techniques against NoC HTs experience rapid development in recent years. However, to the best of our knowledge, there are few standard benchmarks developed for the defense techniques evaluation. To address this issue, in this paper, we design a suite of benchmarks which involves multiple NoCs with different HTs, so that researchers can compare various HT defense methods fairly by making use of them. We first briefly introduce the features of target NoC and its infected modules in our benchmarks, and then, detail the design of our NoC HTs in a one-by-one manner. Finally, we evaluate our benchmarks through extensive simulations and report the circuit cost of NoC HTs in terms of area and power consumption, as well as their effects on NoC performance. Besides, comprehensive experiments, including functional testing and side channel analysis are performed to assess the stealthiness of our HTs.

INDEX TERMS Benchmarks, hardware trojan, network-on-chip.

I. INTRODUCTION

With the development of nano-technology, more and more cores can be integrated into a single chip [1], e.g., CMP (Chip Multi-Processor) and MPSoC (Multi-Processor System-on-Chip), to satisfy the increasing requirement for computational ability from various electronic systems, such as data center, workstation and measuring instrument, etc. NoC (Network-on-Chip) [2], which is proposed to address the scalability, throughput and reliability issues of on-chip communication, has become the preferred communication infrastructure for many/multi-core platforms, as shown in Fig. 1. Thereby, NoC rapidly developed in the last decade, but meanwhile, it attracted growing attention from hardware hackers [3].

In general, the majority of hardware attacks on NoCs come from HTs (Hardware Trojans) [4]. They are malicious modifications on NoC circuit, leading to undesired chip function and/or sensitive information leakage once activated. For example, in reference [5], Song *et al.* describe the procedures for how the routing tables in NoC router can be modified to degrade the system performance. In [6], Reinbrecht *et al.* present how to steal the secret key of AES cryptography from a MPSoC platform by attacking the NoC infrastructure.

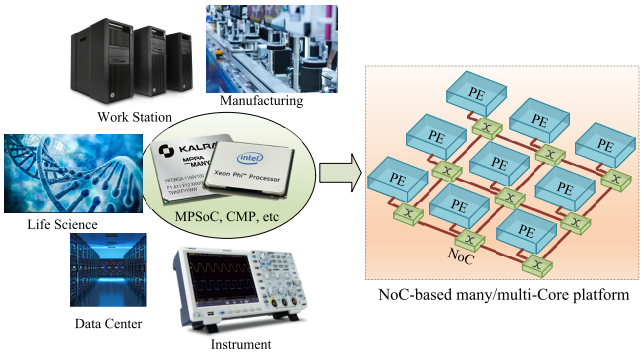


FIGURE 1. NoC-based many/multi-core platforms.

To solve the problems caused by NoC HTs, researches have put many efforts on developing defense techniques, including both HT prevention and HT detection. These works make use of homemade NoC Trojans to evaluate the metrics of their methods since, so far, standard benchmarks which can be used to fairly compare or contrast various NoC HT defense techniques are still lacking. Note that there are some benchmarks frequently used for NoC

The associate editor coordinating the review of this manuscript and approving it for publication was Amjad Mehmood.

TABLE 1. Benchmarks for NoC performance evaluation.

Category	Name	Year	Language
Trace-driven benchmark suites	Netrace [7]	2010	C/C++
	MCSL [8]	2011	SystemC
Model-based benchmark suites	TG [9]	2011	SystemC
	SRBY [10]	2016	C++
Application-based benchmark suites	SPLASH-2 [11]	1995	Fortran,C
	PARSEC [12]	2008	C/C++

performance evaluation, as detailed in TABLE I. However, these benchmarks cannot be used to assess the NoC HT defense techniques because they provide different traffics in software level to drive the target NoC architecture rather than a set of Trojans implemented in hardware level to infect NoC circuit. To break through this dilemma, in this paper, we develop a suite of benchmarks, in which multiple NoCs with different HTs are involved.

The main contributions of this paper can be summarized as follows.

- We design five *trigger* NoC HTs, i.e., livelock Trojan, deadlock Trojan, misrouting Trojan and replay Trojan as well as information leakage Trojan. These Trojans are separately integrated into routers of standard 4×4 mesh NoCs to form our benchmark suite. The suite provides researchers with a level playing field to evaluate the effectiveness of various NoC HTs defense techniques.
- We perform extensive experiments to demonstrate that our HTs (i) have negligible influence on overall area (at most 1.23%) and power (at most 0.315%); (ii) can result in dramatic NoC performance degradation once activated; and (iii) can rarely be triggered through functional testing and detected by analyzing side channel information.

The rest of this paper is as follows. In Section II, we introduce the related works on how to protect NoC against HTs, and in Section III, we develop our benchmarks involving various NoC HTs. The experiments which evaluate our benchmarks in diverse metrics are reported in Section IV. Finally, we conclude our work in Section V.

II. RELATED WORKS

Hardware Trojan has been found over ten years [13]. It greatly challenges the security of chips, calling for powerful HT defense techniques. Hence, in NoC domain, many researches dedicate to HT detection and/or prevention, touching upon both software-based and hardware-based strategies.

A. SOFTWARE-BASED TECHNIQUES

The software-based techniques protect NoCs by exploiting security-aware algorithms and frameworks against HT attacks. They do not make any modification to the original circuits.

For example, Fernandes *et al.* propose a protection technique based on the NoC routing algorithm in [14]. By manipulating the routing of packets, they built multiple security zones, which prioritize communication among paths deemed secure while guaranteeing deadlock freedom. In [15], Kulkarni *et al.* present a real-time anomaly detection framework for many-core router. Under this framework, the hardware Trojan attacks can be found by using machine learning techniques. In [16], Sepulveda *et al.* combine two mechanisms together, random arbitration and adaptive routing, so that the MPSoC can avoid timing side channel attacks launched by Trojans. The experiments reveal that their method is effective to protect the NoC-based MPSoC while increasing the overall performance.

In addition, other software-based techniques are also widely discussed, such as authenticated communication [17], packet dynamic tagging [18], etc.

B. HARDWARE-BASED TECHNIQUES

In contrast to the software-based techniques, hardware-based techniques refer to modification on NoCs, consuming additional logic circuits.

For example, the authors in [19] propose a security enhanced NoC named Gossip, which is able to identify traffic anomalies by integrating a traffic monitor in each router. Results show that their NoC successfully avoid timing attacks with an increase of only 1% in area and 0.8% in power. In [20], Boraten *et al.* introduce Secure Model Checker (SMC), a real-time solution for control logic verification and functional correctness to detect hardware Trojan inducing denial-of-service attacks. The evaluation results show that SMC provides significant security enhancements with only 1.5% power and 1.1% area overhead penalty. To reach the same goal, the authors in [21] and [22] separately present two techniques, termed as link obfuscation and latency auditor, and show their promising merits in protecting NoC against HTs through extensive experiments. To explore the threat posed by a compromised NoC, Ancajas *et al*. propose a three-layer security mechanism. It can be implemented in NoC NI to prevent covert backdoor activation and reduce the chance of a successful side-channel attack [23]. In [24], Yu and Frey exploit transient and permanent error control methods to address HT issues in NoC links, improving the network average latency by up to 44.7% over the rerouting approach.

These works make significant contributions to security side of NoC domain. In this paper, we develop a suite of benchmarks which can be applied to all the aforementioned researches to compare their metrics fairly.

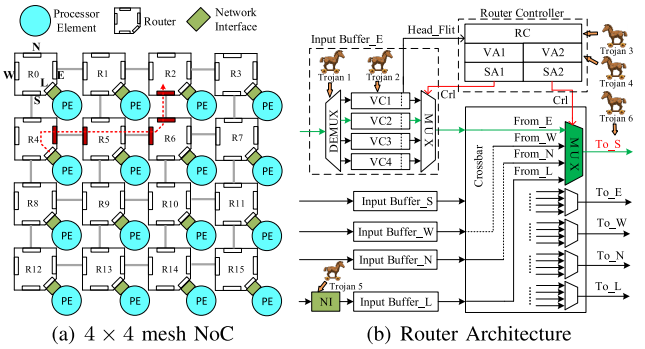
III. OUR BENCHMARKS

Our benchmarks consist of two main parts. One is the target platform, NoC in the paper, and the other refers to hardware Trojans.

A. PRELIMINARY

1) NOC OVERVIEW

Before presenting the design of our Trojans, we briefly introduce the features of our NoC platform. In this paper,



(a) 4 × 4 mesh NoC (b) Router Architecture

FIGURE 2. The overview of CONNECT NoC. (a) 4 × 4 mesh NoC. (b) Router architecture.

we generate a 4 × 4 mesh NoC by using CONNECT tool developed by Carnegie Mellon University [25], as shown in Fig. 2a.[1]

As we can see, when a packet is transmitted from source PE (Processor Element) to its destination, it experiences multiple routers. Each router contains three modules, i.e., Input Buffer, Router Controller and Crossbar. Input Buffer and Crossbar modules belong to the data plane of router, in charge of data storage and transmitting, respectively. In detail, Input Buffer stores the coming packets into their corresponding VCs (Virtual Channels) which are implemented by FIFOs, and Crossbar module transfers packets from input to output ports.

On the other hand, Router Controller manages the working flow of Input Buffer and Crossbar, belonging to the control plane of router. More precisely, when the header flit of a packet arrives at the FIFO header, RC (Routing Computation) module generates a request information for the next hop based on the packet destination and routing algorithm, i.e., X-Y routing in this paper. Then, all requests bid for switching and VA (VC Allocator) decides how to assign VCs of the downstream routers to these requests. Meanwhile, SA (Switch Allocator) which works in parallel with VA picks out the proper packets from VCs and feed them into Crossbar.

2) HARDWARE TROJAN OVERVIEW

Hardware Trojans are malicious modifications on original circuits, which can degrade the performance, change the functionality and/or leak confidential information. As depicted in Fig. 3, all stages in the chip development process (from *specification* to *assembly & package*) can be exploited for hardware Trojan insertion.

In specification phase, the characteristics of the chip, like expected function, size and power requirements, are defined. A Trojan in this phase, for example, can be implemented through tampering the desired functions of the chip. After that, the design phase maps the design to the target technology under multiple constraints, which consists of three stages, RTL (Register Transfer Level) code, netlist and physical layout. These stages are also vulnerable for Trojans insertion. For instance, a rogue in the design house may intentionally hide malicious snippets in the code or add some extra gates to the original netlist. Finally, the manufacturing phase can be divided into two steps: fabrication and assembly & package. Fabrication is to produce desired wafers using masks. Minor changes in the masks may lead to unexpected effects on the final products. Hence, an adversary in the foundry can replace the original masks with the altered masks to realize Trojans in the chip. Then, the step, assembly & package, packages the cut wafers (so-called dies) and assemble the packaged devices on the PCB (Printed Circuit Board) along with many other electronic components. It is possible that the attackers intercept the useful information from the exploitable electromagnetic coupling between the signal on the board and its electromagnetic surroundings, which is introduced by malicious assembly.

In this paper, our hardware Trojans are implemented in RTL level, thus they can be easily implanted to the original design.

B. HARDWARE TROJAN DESIGN

In general, HTs are classified into two categories according to their activation mechanism, namely, *always-on* Trojan and *trigger* Trojan. The *always-on* Trojans work at all time once the infected chips are powered ON, and therefore, they can be easily found by HT detection techniques, e.g., side channel analyzing [29]. In contrast, the *trigger* Trojans are in sleep in most of the chip working time unless receiving a rare, specific wake up signal. In this paper, we design five *trigger* Trojans and separately insert them into link, NI (Network Interface) and various modules of router, i.e., RC, VA, VC and Demux, as shown in Fig. 2b.

Note that 1) NoC is the communication infrastructure in many/multi-core systems. It is in charge of data exchanging among different cores and caches, and therefore, hackers are preferred to attack NoC to degrade its communication performance and/or steal valuable data from it by implanting deadlock, livelock, misrouting, replay and information leakage Trojans. 2) These *trigger* Trojans have two parts, a trigger and a payload. The "trigger" acts as a sensing circuitry. According to the observed signals, it decides when the Trojan "payload" should be activated to perform a specific task. In addition, by adjusting the number of observed bits in signal, we can easily control the activating probability of our hardware Trojans.

1) LIVELOCK TROJAN

Livelock means that packets continuously move in the network but never reach their destinations. The Trojan, as shown in Fig. 4a, monitors the input data at all time and forwards the received data to DEMUX. If the input data does not satisfy the trigger condition, the Trojan remains in non-active state and the payload circuit will not modify the forwarded packets.

[1] Although there are many well-recognized NoC simulators, such as ORION [26], Sniper [27] and Noxim [28], etc., most of them are implemented in high-level language, mainly focusing on the network communication performance. In order to evaluate the hardware overhead of NoC in area and power consumption, in this paper, we adopt CONNECT NoC since it provides synthesizable Verilog code.

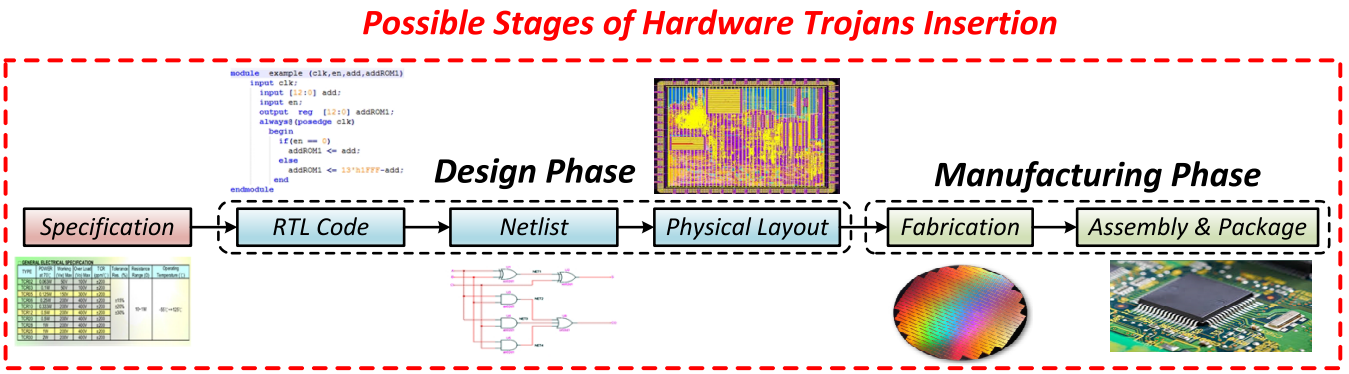


FIGURE 3. Possible stages of hardware Trojans insertion in the chip development process.

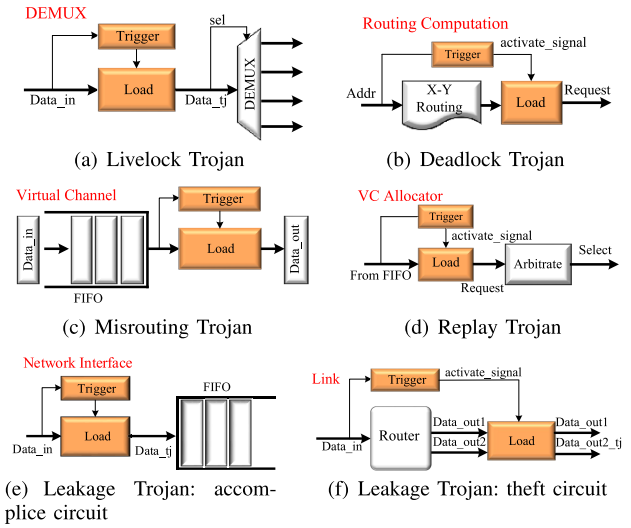


FIGURE 4. NoC hardware Trojan design. (a) Livelock trojan. (b) Deadlock trojan. (c) Misrouting trojan. (d) Replay trojan. (e) Leakage trojan: Accomplice circuit. (f) Leakage trojan: Theft circuit.

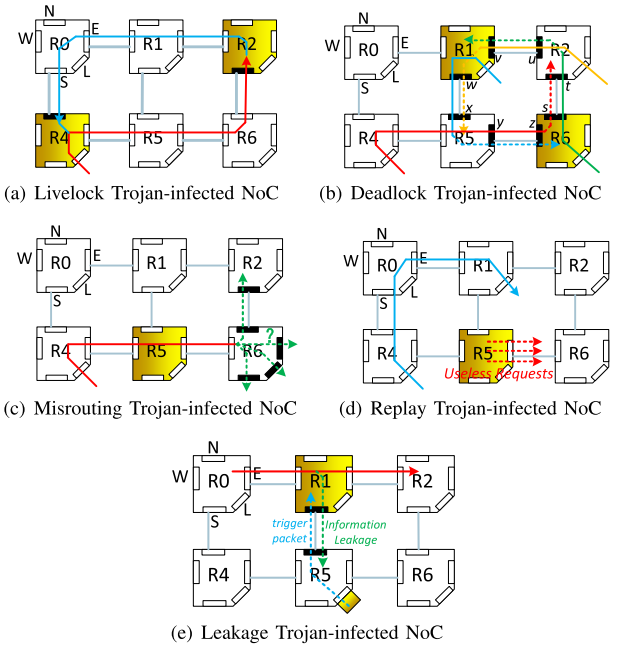


FIGURE 5. Examples of Trojan-infected NoCs. (a) Livelock Trojan-infected NoC. (b) Deadlock Trojan-infected NoC. (c) Misrouting Trojan-infected NoC. (d) Replay Trojan-infected NoC. (e) Leakage Trojan-infected NoC.

Otherwise, the packet destination will be changed to be the other Trojan-infected router. As such, the modified packets will be endlessly transferred between the two Trojan-infected routers, creating a livelock attack.

As shown in Fig. 5a, we insert a livelock Trojan at the input port of DEMUX module for two routers, i.e., *R*2 and *R*4. If the inserted Trojans were dormant, the packets in the NoC would be sent to their destinations naturally. However, once activated, the Trojan hidden in *R*2 would force the current packet being transferred to the another Trojan-infected router *R*4. Similarly, the triggered Trojan in *R*4 would modify the destinations of received packets as *R*2. In this way, packets would fall in the endless loop between the two Trojan-inserted routers.

2) DEADLOCK TROJAN

A deadlock occurs in NoC when a group of packets cannot be further transmitted because they wait for each other to release resources (usually buffers or channels). For example, one packet *A* holds both channels *u* and *v*, but it cannot be further transmitted until the channel *w* is released. At the same time, another packet *B* holds two channels *w* and *x*, while waiting for the release of channel *u*. As such, neither *A* nor *B* can get their required channel, resulting in a deadlock. The diagram of this Trojan is displayed in Fig. 4b. The Trojan trigger monitors the head flits in FIFOs, and judges whether the payload should be activated. The payload provides another routing algorithm which may caused deadlock with the original one. It switches the routing algorithm to the malicious one, if activated.

To implement a deadlock, we insert Trojans in the RC module of two router, i.e., *R*1 and *R*6, as shown in Fig. 5b. There are two normal packets in the NoC. One holds the channels *u* and *v*, waiting the *w* channel for further transmission. The other packet occupies *y* and *z*, attempting to pass through the channel *s*. If the Trojans were in sleep, these two packets can be transferred to their destinations successfully. However, the activated Trojans in *R*1 and *R*6 could change the routing

algorithm and thus make another two new packets obstruct the transmission of the former two packets. In detail, a new packet will occupy channels s and t, requiring the channel u when the other one owns w and x, needing y. Since no one in these four packets wants to make a concession, the further transmission can never be achieved and the deadlock attack is launched successfully.

3) MISROUTING TROJAN

The purpose of the misrouting Trojan is to block the transfer of packets and further leads to the paralysis of entire network.

In general, there are two ways to implement a misrouting Trojan. First, we can modify the packet destination to be an unknown position, such that the Trojan-infected router has no idea how to tackle it, and thereby, keeps the packet in FIFO. Second, we can modify the flow control mechanism of a router, which makes the credit flag always be *full*. As such, the upstream router cannot transfer any packet to the Trojan-infected router since it believes that the Trojan router has no free buffer space to store packet. In the paper, we implement a misrouting Trojan by modifying the packet destination, which is inserted at the output port of buffers. The diagram is given in Fig. 4c. When the Trojan is activated, the destination of packets is replaced with '4bxxxx', so that RC module cannot generate a request information and the congestion happens.

A simple example is given in Fig. 5c, we implant the Trojan in the router $R5$. When packets were fed into the Trojan-infected router $R5$, the working Trojan would replace their destinations with '4bxxxx'. In the next hop, router $R6$ reads these modified packets. However, due to the meaningless '4bxxxx', its RC cannot determine to which direction these packets should be transferred. Hence, these packets can only be stored in $R6$ and cause congestion in the NoC.

4) REPLAY TROJAN

Replay Trojan aims to degrade the network communication performance by generating useless requesting information. For example, even though an input buffer of router is empty, the Trojan still keep requesting for occupying a certain output port, resulting in a waste of bandwidth. In this paper, we implement a replay Trojan and insert it at the input port of VC allocator, as shown in Fig. 4d. The working mechanism of Replay Trojan is similar to the Misrouting Trojan, it observes the head of FIFOs to decide whether activates the Trojan or not. Once the Trojan begins to work, the payload continuously sends the requesting information to VA module. Since CONNECT NoC responses all requests by using polling mechanism, such useless requests can effectively reduce the bandwidth utilization, resulting in the performance losing of NoC.

From Fig. 5d, we can observe that a replay Trojan is inserted into the router $R5$. There is no packets to be transmitted in $R5$. Nevertheless, the Trojan hidden in the router is always sending requesting information to the RC module of $R5$. Note that processing these meaningless requests consumes bandwidth and other resources, which leads to the degradation of NoC communication efficiency.

5) INFORMATION LEAKAGE TROJAN

The objective of information leakage Trojan is to eavesdrop the communication from one router to another. In general, an information leakage Trojan consists of two parts, one accomplice circuit located at a NI and one theft circuit inserted into router links.

The two circuits can work together to steal sensitive information from NoC. As shown in Fig. 4e and 4f, the accomplice circuit in NI monitors the input port of NI buffer at all time, and it will generate a coded sequence of flits to activate the theft circuit when watching a predefined signal from PE (Processing Element). On the other hand, the theft circuit stays asleep and forwards the coming packets directly until its trigger module receives an active command from the accomplice circuit. Once it wakes up, its payload module copies the incoming packets and modifies their destination to be the address where the accomplice circuit is. After that, the duplicated packets are forwarded to the downstream router until they reach their destinations. In this way, the information leakage Trojan can intercept the desired communication in the NoC.

In our benchmark, we insert the accomplice circuit at the network interface corresponding to $R5$ and implant the theft circuit at links connected to $R1$. In detail, the trigger module of theft circuit is located at the link between $R1$ and $R5$, while its payload module is inserted into the links from $R1$ to $R2$ and $R5$. When the accomplice circuit detects a predefined signal from PE, it sends a command to $R5$, waking up the theft circuit. Afterwards, the packets from $R0$ to $R2$ are first duplicated and then transformed to $R5$. As such, the packets sent from $R0$ to $R2$ can be divulged to $R5$ through the information leakage Trojan.

Note that the aforementioned detrimental effects can also be implemented by other circuits which have different architecture and position against our HTs. For example, we could infect the routing computation module to launch a livelock attack in NoC, rather than hijacking the DEMUX module as displayed in Fig. 4a. Hence, the HTs in this paper are not the only circuits corresponding to the detrimental effects, we can design more HTs to enrich our benchmark when necessary.

IV. EXPERIMENTAL RESULTS

We evaluate our benchmarks from three sides. The first one is the circuit overhead of our NoC HTs in area and power consumption, the second one is the HT influence on NoC communication performance, and the final one is the stealthiness of our NoC HTs.

A. POWER AND AREA OVERHEAD

We first implement the proposed five HTs in RTL level by using Verilog code and then separately insert them into a 4×4 CONNECT NoC, in which each input buffer has four virtual channels implemented by 40-bit width and 4-flit

TABLE 2. Power and area cost.

Benchmarks	Power (μW)			Area (μm^2)		
	NoC	HT	RATE	NoC	HT	RATE
Live Lock	82.11	0.26	0.315%	1204676	14995	1.23%
Dead Lock	82.11	0.04	0.049%	1204676	475	0.04%
Misrouting	82.11	0.25	0.304%	1204676	9876	0.81%
Replay	82.11	0.05	0.062%	1204676	731	0.06%
Leakage	82.11	0.21	0.255%	1204676	12890	1.07%

depth FIFOs. Synopsys DC is used to synthesize the area and power consumption of our benchmarks with a 32-nm standard cell library. The system clock frequency is set to be 250MHz. The results are shown in TABLE I. In the table, the first column lists our benchmarks named by the inserted HTs, the 2nd-4th columns report their corresponding power consumption in μW and the 5th-7th columns detail their area cost in μm^2. RATE means the percent of HT cost in a benchmark, which can be calculated by $HT/(NoC + HT)$.

From the table, we can find that the five HTs need little logic resources to be implemented, since their power consumption ranges from 0.04 μW to 0.26 μW. Furthermore, each Trojan only takes small percentage, from 0.049% (Deadlock) to 0.315% (Livelock) when compared to the total power consumption of benchmark. From the perspective of area cost, we can draw the same conclusion. In detail, the five HTs separately occupy area from 475 μm^2 to 14995 μm^2, which takes small percentage of the whole chip, i.e., from 0.04% to 1.23%.

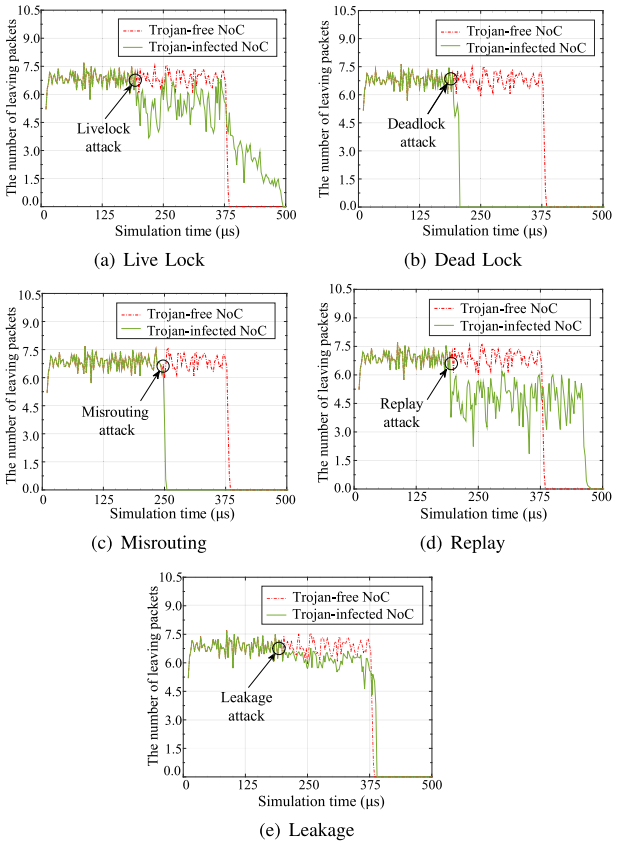


FIGURE 6. Trojan influence on NoC performance. (a) Live lock. (b) Dead lock. (c) Misrouting. (d) Replay. (e) Leakage.

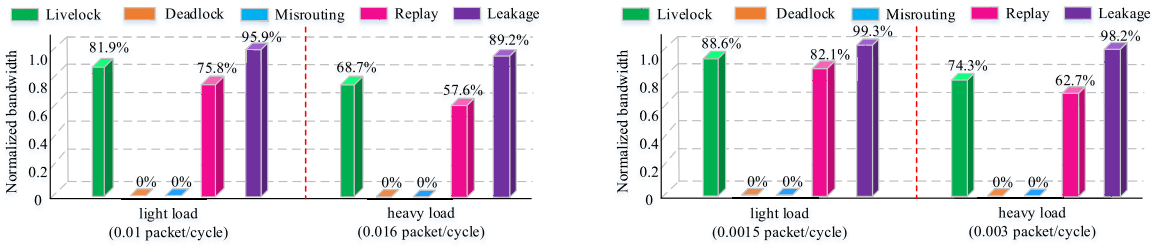
B. NETWORK PERFORMANCE EVALUATION

We display the influence of HTs on NoC communication performance before/after they are activated. To reach this goal, we design a complete peripheral module in Verilog code which can both inject packet into NoC with a given rate (in Possion distribution) and count the number of leaving packets from NoC for a predefined time duration. In this paper, we set the packet injection rate to be 0.028 packets/cycle, which is the critical point for the 4 × 4 CONNECT NoC. Then, we perform RTL level simulation by using Modelsim tool, and collect the leaving packets for each 2500 ns during the whole simulation process. For each benchmark, we activate the inserted HT after the Trojan-infected NoC works for about 200 μs. In addition, we also observe the communication performance from a Trojan-free NoC for comparison purpose. Note that critical point is defined as the packet injection rate that doubles the average network communication latency compared to NoC under loads without contention. It is the boundary between NoC heavy loads and light loads, and therefore, often used for NoC performance evaluation [2][30].

The experimental results are shown in Fig. 6. From the figure, we can find that the Trojan-infected NoCs have the same performance as the Trojan-free ones before HTs are activated. However, once the HTs wake up, the performance of all Trojan-infected NoCs will be degraded. More precisely, the Deadlock and Misrouting Trojans almost use up all the NoC bandwidth once they are activated, leading to zero leaving packets immediately. On the other hand, the activate Livelock, Replay and information leakage Trojans only waste a fraction of NoC bandwidth, which makes the rate of leaving packets reduced in the rest simulation time.

To evaluate the effect of Trojans at different NoC loads and sizes, we separately insert five Trojans, i.e., livelock, deadlock, misrouting, replay and information leakage Trojans, into 8 × 8 and 16 × 16 mesh NoCs. Then, we activate these Trojans under light and heavy network loads, and in Fig. 7, we report the NoC bandwidth normalized to the benign NoCs.

From Fig. 7, we can draw the following conclusions. First, the activated deadlock and misrouting Trojans seriously threat the NoC communication performance. They consume all bandwidth for both large and small scale NoCs, no matter which load they work on. Second, the livelock, relay and leakage Trojans have different influences on NoC performance with the variation of network sizes and loads. In general, they lead to more performance losing for the small scale NoC under heavy load. For example, when the livelock Trojan is activated, the 8 × 8 NoC under heavy load, i.e., 0.016 packet/cycle, remains only 68.7% bandwidth. However, for the 8 × 8 NoC under light load and the 16 × 16 NoC under heavy load, the reserved bandwidths achieve 81.9% and 74.3%, respectively.



(a) 8 × 8 mesh NoC

(b) 16 × 16 mesh NoC

FIGURE 7. The effect of Trojans at different NoC loads and sizes. (a) 8 × 8 mesh NoC. (b) 16 × 16 mesh NoC.

C. TROJANS STEALTHINESS EVALUATION

We evaluate the stealthiness of our NoC Trojans through different methods. In detail, we first perform function testing on our benchmarks to assess how hard our Trojans could be triggered. After the assessment, we analyze the electromagnetic emissions from the running design through the side channel technique proposed in [31], attempting to find out those Trojans.

1) FUNCTIONAL TESTING

In this experiment, we carefully design a peripheral packet injector in Verilog to feed data packets to our benchmarks. An input signal controls whether the injector is turned on or not. Three kinds of traffic patterns, namely, uniform, hotspot and bit-reverse, can be applied on the experimental NoCs [2], and NoCs here work in the uniform traffic pattern. According to the pattern, the peripheral injector will generate 10,000,000 packets, 32 40-bit flits in each packet, and inject them to the NoC with a Poisson distribution. Note that we embed a pseudo-random number generator in the injector to assign the random values for each packet, to simulate the real-world scenario. By using the NoC, these packets can be transferred to their destinations with X-Y routing.

In the testing process for each benchmark, we first implement the benchmark along with the peripheral injector on an Artix-7 FPGA. Then we turn on the injector by sending it a specific signal. After that, the NoC starts to transfer the fed packets from the injector to their destinations. In the process, we employ the Xilinx logic analyzer ChipScope to observe the activate signal of the Trojan. If the activate signal value changes, it means that the trojan can be triggered using functional testing.

From the second column of TABLE II, we can observe that no Trojans can be activated under the stimulation of numerous testing vectors (for each benchmark, the number of vectors equals to the generated flits $10{,}000{,}000 \times 32 = 320{,}000{,}000$). The experimental results can be explained by two reasons. First, the routers which can be experienced by certain packet are determined by the destination and the routing algorithm. It is completely possible that a packet could have to trigger the Trojan but it can never reach the HT-infected routers, so that the NoC can be free from the potential attack. On the other hand, even though all packets pass through the infected routers, the Trojans can still be hardly activated since they can be activated only when the values of the current flit exactly match the pre-defined values. It means that the activating probability of these Trojans is extremely low ($\frac{1}{2^{40}}$). In this case, the probability that the Trojan is activated through 320,000,000 random vectors can be approximately calculated as $\frac{3}{10000}$. The reasonable results demonstrate that our Trojans can rarely be triggered, and thus are resistant to functional testing.

TABLE 3. Detection results of different methods on our benchmarks.

Benchmarks	Methods	
	Functional Testing	Side Channel Analysis
Live Lock	×	×
Dead Lock	×	×
Misrouting	×	×
Replay	×	×
Leakage	×	×

× means that the Trojan cannot be detected using the method.

2) SIDE CHANNEL ANALYSIS

From the Section IV-C 1), we can claim that our Trojans can hardly be triggered. To further prove the stealthiness of our Trojans, we detect these dormant Trojans by using the side channel technique, namely, electromagnetic (EM) detection [31].

In this experiment, the 4 × 4 CONNECT mesh NoC serves as our golden design, and HT-infected NoCs are experimental targets. One Xilinx Artix-7 FPGA is selected as our platform for design implementation. We place the experimental FPGA on a X-Y-Z positioning system and use a near-field probe (Langer ICR HH500-6) above the target FPGA to measure its EM emissions. To obtain a better signal strength, we open the package of the target FPGA chip and put the probe close to the circuitry.

During the experiment, we first implement the golden design on the FPGA and collect its EM emanations. Particularly, we represent its area as a matrix of 17 rows and 21 columns and perform 3000 measurements at each point to get an average value. In this way, we can get a set of data showing the EM distribution of the golden design. With the same method, the EM emissions of the HT-infected designs can also be collected. Then we subtract the data of the golden design to the data of HT-infected ones point-by-point. If there

exist significant differences, we can claim that the dormant HT can be detected using the side channel technique.

From the third column of TABLE II, we can find that no Trojans have been found out through the EM detection. It is understandable since our Trojans only occupy at most 1.23% of the whole area. However, according to [31], the EM detection requires an applicable HT scale being at least 1.3% of the whole circuit. If the target HTs are smaller than the scale, the technique will be hard to deal with them.

V. CONCLUSION

In this paper, we propose a suite of benchmarks which contain different hardware Trojans in a standard CONNECT NoC, namely, deadlock, livelock, misrouting, replay and leakage. After briefly introducing the features of CONNECT NoC and its infected modules in our benchmarks, we separately detail the design of our NoC HTs. The simulation results reveal that our HTs in the NoC has the characteristics of low power consumption and small area, and they can effectively reduce the communication performance of NoCs once activated. Besides, the extensive experiments demonstrate that our HTs are hard to be triggered and detected. Therefore, our benchmarks are meaningful for the researches on NoC security, since they can be used to fairly compare or contrast any new HT defense technique with existing ones.

Besides the five Trojans, there are some other kinds of NoC Trojan as well. For example, a malicious circuit could be designed to power router OFF or launch DoS (Denial-of-Service) attack at links when it is activated. Also, a multi-stage ring oscillator could be placed among routers as an *always-on* Trojan. It produces thermal at all working time, triggering the chip thermal threshold to degrade the NoC frequency. In the future, we plan to add more Trojans to our repository and develop effective NoC Trojan detection techniques with them.

REFERENCES

[1] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Proc. 38th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2011, pp. 365–376.

[2] J. Wang, S. Guo, Z. Chen, Y. Li, and Z. Lu, "A new parallel CODEC technique for CDMA NoCs," *IEEE Trans. Ind. Electron.*, vol. 65, no. 8, pp. 6527–6537, Dec. 2018.

[3] L. Fiorin, C. Silvano, and M. Sami, "Security aspects in networks-on-chips: Overview and proposals for secure implementations," in *Proc. 10th Euromicro Conf. Digit. Syst. Design Archit., Methods Tools*, Aug. 2007, pp. 539–542.

[4] M. Tehranipoor and F. Koushanfar, "A survey of hardware Trojan taxonomy and detection," *IEEE Design Test Comput.*, vol. 27, no. 1, pp. 10–25, Feb. 2010.

[5] W. Song, J. Kim, J.-W. Lee, and D. Abts, "Security vulnerability in processor-interconnect router design," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 358–368.

[6] C. Reinbrecht, A. Susin, L. Bossuet, G. Sigl, and J. Sepúlveda, "Side channel attack on NoC-based MPSoCs are practical: NoC prime+probe attack," in *Proc. 29th Symp. Integr. Circuits Syst. Design*, Aug. 2016, pp. 1–6.

[7] J. Hestness, B. Grot, and S. W. Keckler, "Netrace: Dependency-driven trace-based network-on-chip simulation," in *Proc. 3rd Int. Workshop Netw. Chip Archit.*, 2010, pp. 31–36.

[8] W. Liu, J. Xu, X. Wu, Y. Ye, X. Wang, W. Zhang, M. Nikdast, and Z. Wang, "A NoC traffic suite based on real applications," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, Jul. 2011, pp. 66–71.

[9] E. Pekkarinen, L. Lehtonen, E. Salminen, and T. D. Hämäläinen, "A set of traffic models for network-on-chip benchmarking," in *Proc. Int. Symp. Syst. Chip (SoC)*, Oct. 2011, pp. 78–81.

[10] Y. Xue and P. Bogdan, "Scalable and realistic benchmark synthesis for efficient NoC performance evaluation: A complex network analysis approach," in *Proc. Int. Conf. Hardw./Softw. Codesign Syst. Synthesis (CODES+ISSS)*, Oct. 2016, pp. 1–10.

[11] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: Characterization and architectural implications," in *Proc. 17th Int. Conf. Parallel Archit. Compilation Techn.*, 2008, pp. 72–81.

[12] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: Characterization and methodological considerations," *ACM SIGARCH Comput. Archit. News*, vol. 23, no. 2, pp. 24–36, 1995.

[13] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using IC fingerprinting," in *Proc. IEEE Symp. Secur. Privacy*, May 2007, pp. 296–310.

[14] R. Fernandes, C. Marcon, R. Cataldo, J. Silveira, G. Sigl, and J. Sepúlveda, "A security aware routing approach for NoC-based MPSoCs," in *Proc. 29th Symp. Integr. Circuits Syst. Design (SBCCI)*, Aug. 2016, pp. 1–6.

[15] A. Kulkarni, Y. Pino, M. French, and T. Mohsenin, "Real-time anomaly detection framework for many-core router through machine-learning techniques," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 1, p. 10, 2016.

[16] M. J. Sepulveda, J. P. Diguet, M. Strum, and G. Gogniat, "NoC-based protection for SoC time-driven attacks," *IEEE Embedded Syst. Lett.*, vol. 7, no. 1, pp. 7–10, Mar. 2015.

[17] H. K. Kapoor, G. B. Rao, S. Arshi, and G. Trivedi, "A security framework for NoC using authenticated encryption and session keys," *Circuits, Syst., Signal Process.*, vol. 32, no. 6, pp. 2605–2622, 2013.

[18] M. Hussain and H. Guo, "Packet leak detection on hardware-trojan infected NoCs for MPSoC systems," in *Proc. Int. Conf. Cryptogr., Secur. Privacy*, 2017, pp. 85–90.

[19] C. Reinbrecht, A. Susin, L. Bossuet, and J. Sepúlveda, "Gossip NoC—Avoiding timing side-channel attacks through traffic management," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2016, pp. 601–606.

[20] T. Boraten, D. DiTomaso, and A. K. Kodi, "Secure model checkers for network-on-chip (NoC) architectures," in *Proc. Int. Great Lakes Symp. VLSI*, May 2016, pp. 45–50.

[21] T. Boraten and A. Kodi, "Mitigation of Hardware Trojan based denial-of-service attack for secure NoCs," *J. Parallel Distrib. Comput.*, vol. 111, pp. 24–38, Jan. 2018.

[22] J. S. Rajesh, D. M. Ancajas, K. Chakraborty, and S. Roy, "Runtime detection of a bandwidth denial attack from a rogue network-on-chip," in *Proc. 9th Int. Symp. Netw.-Chip*, 2015, p. 8.

[23] D. M. Ancajas, K. Chakraborty, and S. Roy, "Fort-NoCs: Mitigating the threat of a compromised NoC," in *Proc. 51st Annu. Design Autom. Conf.*, 2014, pp. 1–6.

[24] 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. (DFTS)*, Oct. 2013, pp. 266–271.

[25] M. K. Papamichael and J. C. Hoe, "The CONNECT network-on-chip generator," *Computer*, vol. 48, no. 12, pp. 72–79, Dec. 2015.

[26] A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi, "Orion 2.0: A power-area simulator for interconnection networks," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 1, pp. 191–196, Jan. 2012.

[27] T. E. Carlson, W. Heirmant, and L. Eeckhout, "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2011, pp. 1–12.

[28] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti, "Cycle-accurate network on chip simulation with noxim," *ACM Trans. Model. Comput. Simul.*, vol. 27, no. 1, p. 4, 2016.

[29] S. Narasimhan, D. Du, R. S. Chakraborty, S. Paul, F. Wolff, C. Papachristou, K. Roy, and S. Bhunia, "Multiple-parameter side-channel analysis: A non-invasive hardware Trojan detection approach," in *Proc. IEEE Int. Symp. Hardw.-Oriented Secur. Trust (HOST)*, Jun. 2010, pp. 13–18.

[30] Z. Lu and Y. Yao, "Dynamic traffic regulation in NoC-based systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 2, pp. 556–569, Jul. 2016.

[31] J. Balasch, B. Gierlichs, and I. Verbauwhede, "Electromagnetic circuit fingerprints for hardware trojan detection," in *Proc. IEEE Int. Symp. Electromagn. Compat. (EMC)*, Aug. 2015, pp. 246–251.