IET Computers & Digital Techniques

Review Article

IET Journals
The Institution of Engineering and Technology

Ten years of hardware Trojans: a survey from the attacker's perspective

ISSN 1751-8601
Received on 16th February 2020
Revised 26th June 2020
Accepted on 3rd July 2020
E-First on 30th September 2020
doi: 10.1049/iet-cdt.2020.0041
www.ietdl.org

Mingfu Xue[1] ✉, Chongyan Gu[2], Weiqiang Liu[3], Shichao Yu[2], Máire O'Neill[2]

[1]College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, People's Republic of China
[2]Center for Secure Information Technologies, Queen's University Belfast, Belfast, UK
[3]College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, People's Republic of China
✉ *E-mail: mingfu.xue@nuaa.edu.cn*

Abstract: Hardware Trojan detection techniques have been studied extensively. However, to develop reliable and effective defenses, it is important to figure out how hardware Trojans are implemented in practical scenarios. The authors attempt to make a review of the hardware Trojan design and implementations in the last decade and also provide an outlook. Unlike all previous surveys that discuss Trojans from the defender's perspective, for the first time, the authors study the Trojans from the attacker's perspective, focusing on the attacker's methods, capabilities, and challenges when the attacker designs and implements a hardware Trojan. First, the authors present adversarial models in terms of the adversary's methods, adversary's capabilities, and adversary's challenges in seven practical hardware Trojan implementation scenarios: in-house design team attacks, third-party intellectual property vendor attacks, computer-aided design tools attacks, fabrication stage attacks, testing stage attacks, distribution stage attacks, and field-programmable gate array Trojan attacks. Second, the authors analyse the hardware Trojan implementation methods under each adversarial model in terms of seven aspects/metrics: hardware Trojan attack scenarios, the attacker's motivation, feasibility, detectability (anti-detection capability), protection and prevention suggestions for the designer, overhead analysis, and case studies of Trojan implementations. Finally, future directions on hardware Trojan attacks and defenses are also discussed.

1 Introduction

In the last decade, the malicious modifications of integrated circuits (ICs), also referred to as hardware Trojans (HTs), have become emerging security concerns in the IC industry. ICs that contain HTs can cause malfunction, leakage of confidential information, or lead to other disastrous consequences. Therefore, HT has been a matter of concern for the industry, academia, government, and military [1–4].

Since the first research on HT published in 2007 by Agrawal *et al.* [5], HTs have been developed for more than ten years. A lot of research has been conducted on detecting HTs. However, there has been very little research on the implementation of HTs in practice. To develop reliable HT detection and defense techniques, it is necessary to understand the feasibility of inserting HTs in practical implementations [6]. More specifically, how stealthy HTs can be inserted into a target circuit, how feasible is HT for a specific application model, and what are the challenges to implementing such HTs [7]. This remains a field that has received relatively little attention in the research community where most HTs referred to in the literature are small- or medium-sized circuits added at register transfer level (RTL) during the IC design flow [7].

In this paper, we attempt to make a review of the HT implementations in the last decade and also make an outlook. In particular, unlike all previous surveys that discuss Trojans from the defender's perspective, for the first time, we will discuss the Trojans from the attacker's perspective, focusing on the attacker's methods, feasibility, anti-detection capability, and challenges when designing and implementing a HT. As Chinese strategist Sun Tzu said, 'If you know yourself and the enemy, you will never lose a battle', (Sun Tzu, The Art of War, ancient Chinese military philosophy book). Discussing Trojans from an attacker's perspective can give readers a clear understanding of an attacker's considerations when implementing a HT, including advantages and deficiencies of the attacker, trade-offs, and the methods that the attacker can take. This can hopefully help designers better understand the Trojan insertion, and provide guidelines for the defenders to design better detection and defense techniques.

Particularly, in this survey, we want to explore the following questions:

Q1: HT techniques have been developed for more than ten years. What attack methods do the attackers at various stages have?
Q2: What are the attackers' capabilities at various abstraction levels to launch the attacks?
Q3: With the continuous development of Trojan detection techniques, what are the challenges and difficulties faced by the attackers?
Q4: How attackers could overcome these state-of-the-art Trojan detection techniques and what kind of new detection technique is required?

In addition, this paper will present several new insights and assumptions for the first time. For example, in the literature, it is usually assumed that the HT problem was discussed from the perspective of the copyright owner or the designer, i.e. the copyright owner is assumed to be trustworthy and the HT was inserted in the untrustworthy design and fabrication processes. However, when returning to the essential definition of HTs, HT is a Trojan/backdoor in the form of hardware, which is not limited to the copyright owner's perspective. Moreover, the number of users is much larger than the number of the copyright owners, and the users are usually in weaker positions than the copyright owners. Therefore, it is also necessary to consider the Trojans from the user's perspective. For the users, the hardware Trojan/backdoor implanted by the copyright owners or designers in the device can also be regarded as an HT, such as some super privileged structures which can be remotely controlled by the copyright owner, the hardware that transmits user's data back to the copyright owner, or the hardware that can record the user's keystrokes, and so on.

There have already been some reported accidents of potential HT attacks to gain control of devices, steal secret information, or

even destroy a system. In September 2007, Israel launched a successful airstrike on a nuclear reactor in Syria, while Syria's advanced air defense system did not respond throughout the operation [8]. In 2008, Adee [8] speculated that Syria's air defense system had been deactivated by a built-in kill switch, which could be accessed and activated remotely. Since the practical HTs used in industrial fields and military are often highly confidential, researchers cannot accurately determine whether they are HTs and their implementation details. However, it still shows the concerns of various communities about the destructive power of HTs. In 2016, Yang *et al.* [9] proposed a small malicious HT, named A2, where they implemented a privilege escalation attack in the OR1200 processor by running a set of seemingly harmless commands. Such lightweight analog malicious backdoors are extremely difficult to detect. In January 2018, the Free Software Foundation revealed that Intel computers have a built-in subsystem, called the Intel Management Engine (ME), which can take full control over the computer, and even has access to the main memory [10, 11]. The ME structure can be a serious threat to the users' privacy and security. However, users do not have the ability to audit, examine, or disable it [10, 11]. From the user's point of view, this could also be considered as an HT.

To date, several review and survey papers on HT detection techniques or HT taxonomy have been published. Bhunia *et al.* [1] analysed the threats of HT attacks, Trojan models and classifications, and protection approaches. They mainly focused on various defense techniques against HTs, including HT detection techniques, design-for-security (DFS) approaches, and runtime monitoring techniques. Tehranipoor and Koushanfar [2] presented a classification of HTs and a survey of Trojan detection techniques. In particular, they presented existing detection mechanisms and DFS methodologies. Chakraborty *et al.* [3] presented a Trojan taxonomy and a review of state-of-the-art HT detection techniques. Rostami *et al.* [12] systematised various hardware security-related attacks, including HTs, reverse engineering (RE), IC overbuilding and intellectual property (IP) piracy, side-channel analysis (SCA) etc. Jacob *et al.* [13] reviewed HT vulnerabilities in the IC's life cycle and HT detection techniques. Karri *et al.* [14] proposed a Trojan taxonomy in terms of the activation mechanism, effects, abstraction level, insertion phase, and location. The above surveys all focus on HT detection or HT taxonomy, and are published before 2014, while a large number of HT works that have emerged in the past six years are not included.

Different from all existing surveys, this paper presents a survey of HT design and implementation based on practical attack scenarios from an attacker's perspective. The differences between this survey and these existing review/survey papers are summarised as follows:

(i) The HT design and implementation methods are systematically studied and analysed, focusing on the attacker's insertion methods, capabilities, evading detection techniques, and challenges when the attacker designs and implements an HT. To the best of the authors' knowledge, this is the first survey of HT design and implementation methods from an attacker's perspective, instead of HT detection techniques from the defender's perspective.

(ii) We present adversarial models that show *adversary's methods*, *adversary's capabilities* and *adversary's challenges* to insert HT into a chip in seven practical HT implementation scenarios: in-house design team attacks, third-party IP (3PIP) vendor attacks, computer-aided design (CAD) tools attacks, fabrication stage attacks, testing stage attacks, distribution stage attacks, and field-programmable gate array (FPGA) Trojan attacks. Note that the contribution of this paper is not to provide a new HT taxonomy. Trojan taxonomies have been widely mentioned in existing review literature. The goal of this paper is to analyse the attacker's considerations during Trojan insertion in various practical scenarios, including the technical options, advantages and disadvantages, trade-offs, anti-detection capabilities, and so on.

(iii) HT design and implementation methods under each adversarial model are reviewed in terms of seven aspects/metrics: HT attack scenario, motivation, feasibility, detectability (anti-detection capability), protection and prevention suggestions, overhead, and case studies. The feasibility and detectability are two main concerns from the attackers' perspective. The protection and prevention suggestions and overhead are two metrics from the defenders' perspective. Note that the existing HT literature, including survey works, mostly focus on HT detection and defenses. Therefore, in this paper, we do not discuss the HT detection and defense techniques in detail, but only give brief suggestions for Trojan detection. Instead, we will discuss the feasibility and anti-detection capability in detail when inserting Trojans from the attacker's perspective.

(iv) Future potential directions on HT designs and defenses are also discussed, including HT benchmarks and evaluation methods, machine learning-based Trojan detection methods and HTs targeting machine learning models, attacks and defenses from chips to complex systems, universal Trojan and automatic Trojan insertion versus automatic Trojan (IC vulnerability) analysis tools, multi-stage HT attacks and defenses, split manufacturing, low overhead runtime HT monitoring techniques, logic obfuscation for HT prevention, and FPGA Trojan attacks and defenses.

(v) This paper presents several new insights and assumptions for the first time. On the one hand, researchers should not only consider the HTs implanted during the untrustworthy design and fabrication stages from the copyright owner's perspective, but should also consider the Trojans inserted by the copyright owner from the user's perspective. On the other hand, existing works usually only consider Trojans to be inserted in the design stage, CAD tools, and fabrication stage. In this paper, for the first time, we also systematically discuss the HT attacks in the testing stage and distribution stage. Moreover, the emerging FPGA Trojan attacks are also systematically discussed.

The rest of this paper is organised as follows. The attack models are described in Section 2. In-house design team attacks are analysed in Section 3. 3PIP vendor attacks are presented in Section 4. CAD tools attacks are described in Section 5. Fabrication stage attacks are presented in Section 6. Testing stage attacks are described in Section 7. Distribution stage attacks are analysed in Section 8. FPGA Trojans are presented in Section 9. Future directions are discussed in Section 10. Finally, conclusions are provided in Section 11.

2 Attack models: adversary's methods, capabilities, and challenges

In this section, we will present the attack models in terms of the adversary's methods, adversary's capabilities, and adversary's challenges in seven practical HT implementation scenarios.

A malicious attacker in any stage of the IC supply chain can insert HTs. The most common concern is that HTs can be inserted during fabrication by untrusted foundries. A malicious designer in the IC design team could also manipulate the design and have the flexibility to implement various HTs. Similarly, 3PIP core is another possible source of HTs [15]. Other entities, e.g. CAD tool vendors, IC vendors, and users, although have less chance to insert an HT, but are still feasible to implement HT attacks. HT design and implementation methods are diverse, e.g. an attacker can design an HT based on the desired attack function, triggering mechanism, insertion stage etc.

As the HT design and implementation methods significantly depend on practical application scenarios and the attackers' intentions, in this paper, we present adversarial models in terms of adversary's methods, adversary's capabilities, and adversary's challenges in different HT implementation scenarios, as shown in Fig. 1. Related works usually consider the testing phase to be trusted, while in this paper, we consider that the testing phase may also be untrustworthy. Strictly speaking, in the testing phase, untrusted testing organisations are not able to insert Trojans, but can only collude with the malicious factory or designers to hide the inserted HTs, i.e. making the HTs evade detection. Similarly, it is generally considered that the distributor cannot insert an HT because the distributor is usually unaware of the design. However, a distribution stage attacker can RE a chip to pirate the chip. The attacker can also directly replace the circuit with a Trojan-inserted

circuit during transportation. Therefore, in this paper, we also discuss the attackers from the testing stage and distribution stage. Specifically, we divide the practical HT implementation scenarios into seven different adversarial models: (i) in-house design team attacks; (ii) 3PIP vendor attacks; (iii) CAD tools attacks; (iv) fabrication stage attacks; (v) testing stage attacks; (vi) distribution stage attacks; and (vii) FPGA Trojan attacks. Moreover, we analyse the HT implementation methods under each adversarial model in terms of the following seven aspects/metrics:

(i) *HT attack scenario*: a description of the HT attack scenario, including the HT types, trigger mechanisms, payloads etc.
(ii) *Motivation*: the motivation of an attacker, including the malicious functions that an attacker wants to achieve.
(iii) *Feasibility*: the practicality of the attacks, including the resources available for an attacker, the HT design methods that an attacker can adopt. Similar to cryptography and cryptanalysis, the attacker is assumed to have significant resources, but they are restricted by the rule that the benefit from the Trojan attack should exceed the resources expended [16]. The HTs should also be practical and effective under practical scenarios and be easy to control so that an attacker can employ them to perform attacks easily.
(iv) *Detectability*: anti-detection capability of the Trojan, i.e. how to evade the state-of-the-art defenses from the attacker's perspective. In other words, the detection mechanisms available for the described HT and how likely the HT will be detected.
(v) *Protection and prevention suggestions*: guidelines for designers about protection and prevention, including challenges and opportunities from the designer's perspective, suggestions that would help designers to protect the circuits better against Trojan insertions, and how the attack models will affect the future secure hardware design. As mentioned in Section 1, since most of the existing works have discussed HT detection techniques, in this paper, we do not discuss the HT detection techniques in detail, but only give brief suggestions for the Trojan detection (referred to as protection and prevention suggestions). Instead, we will discuss in detail the anti-detection capability of an HT and the attacker's considerations of evading detection from the attacker's perspective (referred to as detectability).
(vi) *Overhead*: cost for Trojan detection from the defender's perspective, in terms of power, area, and performance.
(vii) *Case studies*: examples of HT design and implementations.

In the following sections, HT design and implementation methods are reviewed and analysed under the above seven practical HT implementation scenarios, respectively, in terms of the above seven aspects. Particularly, Fig. 2 presents the HT attack scenario, motivation, available resources, feasibility, detectability (anti-detection capability), case studies from the attacker's perspective, and the protection and prevention suggestions, overhead from the defender's perspective under different attack models, which will be discussed in the following sections.

3 In-house design team attacks

HT attack scenario: This attack model is the one that most commonly referred to in the literature. Rogue designers in an outsourced or in-house design team can easily implement stealthy malicious modifications in the RTL design since the attackers can get the source files and codes, as shown in Fig. 2. Trojans inserted by the malicious designer can implement any possible payloads with various trigger methods.
Motivation: The attacker in the design stage who insert an HT into the IC may want to steal confidential information from the deployed ICs, or cause malfunction of the ICs.
Feasibility: The attackers can manipulate the circuit with high flexibility to implement any malicious functions. The trigger is expected to be undetectable by functional tests. A feasible approach is to use a specifically designed input sequence, e.g. an abnormal condition, or a rare event. However, a trigger that relies on physical access may be restricted in practical applications. Therefore, some internal signals, e.g. a counter, a specified temperature, or voltage, can be used as an activation mechanism for the Trojan, such as the RS232-T200 HT [49]. Another type of trigger, which is more aggressive, configures the Trojan as always-on. In this case, the payload of the Trojan is required to be hidden, e.g. sending secret information undetected by functional tests, such as the Advanced Encryption Standard (AES)-T200 HT [49]. However, the always-on Trojans may introduce high-power consumption, which could be easily discovered by SCA methods

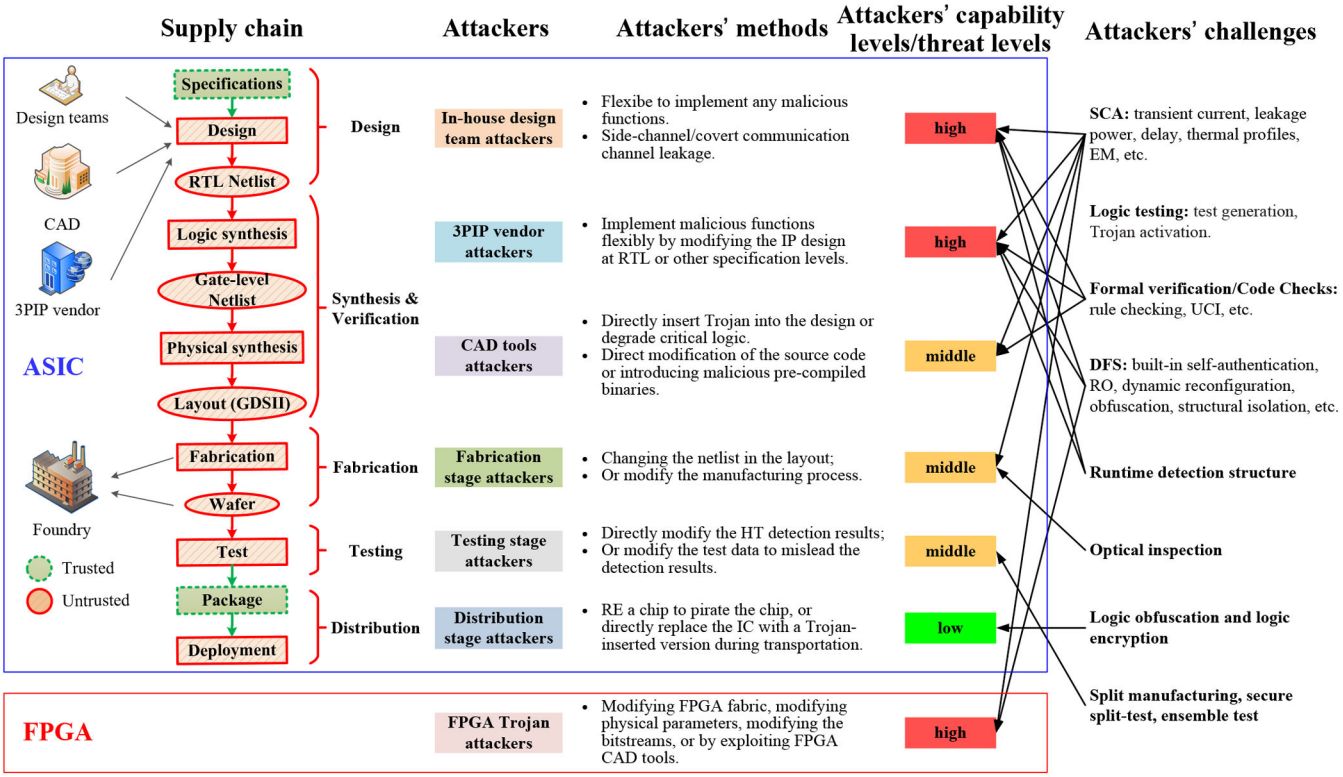

Fig. 1 *HT attack models in terms of the adversary's methods, adversary's capabilities, and adversary's challenges*

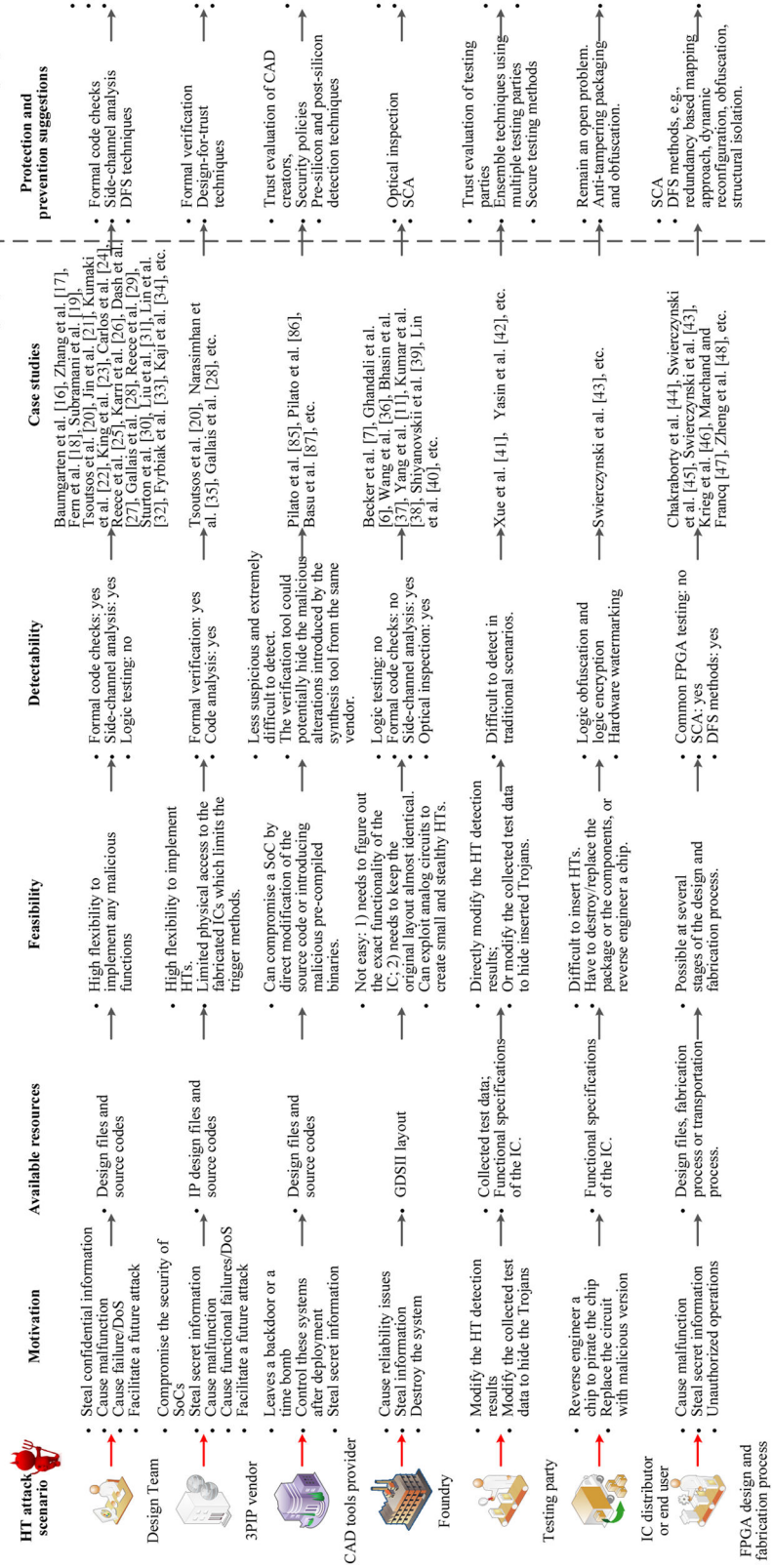

Fig. 2 *HT attack scenario, motivation, available resources, feasibility, detectability (anti-detection capability), case studies under different attack models from the attacker's perspective, and protection and prevention suggestions, overhead from the defender's perspective*

[21]. The reasons are as follows. In general, HT is triggered by rare events to evade the detection of defense techniques. As a result, the HT is latent during most of the time. It does not affect the logic values of the circuit nodes, and rarely generates transition

activities, so the power consumption introduced by the Trojan is very low. However, for always-on Trojan, on the one hand, the payload usually does not directly affect the digital value of the circuit node, so as not to be easily detected by logic tests. On the other hand, it has no triggering conditions and is always on, so its circuit transition activities will be relatively high. Thus, it will be easily detected by the SCA method based on dynamic current or power consumption. Note that, as a special case, parametric Trojans can also be considered as always-on, which does not necessarily lead to high-power consumption. In conclusion, the design stage attack has high feasibility, good practicality, and is easy to implement.

Detectability: First, we discuss the available detection methods from the designer's point of view. The insertion of HTs in the RTL can be revealed by formal code checks, which requires a comprehensive security policy to counter all possible threats. Using SCA, the impacts of Trojans on circuits' delay, leakage power, transient current, thermal profiles, electromagnetic emanation (EM) etc., can be characterised for HT detection [50–57]. In addition to these traditional side-channel signals (power, delay, EM etc.), emerging side-channel signals also include impedance [58], backscattering [59], channel noise in wireless channels [19] etc. The traditional functional test fails to detect HTs, therefore, a few HT detection works have also been proposed to generate test patterns that target rarely activated nodes or events in a circuit [60–63]. However, the huge number of gates and states in modern ICs restrict the accuracy and scalability of these methods.

Second, we discuss how to evade detection from an attacker's perspective. A malicious designer would normally have expertise in IC design. Thus, they could insert optimised HT designs that balance the area and power overhead. Since RTL modification could affect all taped-out ICs, a golden model may not exist for use in timing and power analysis-based detection methods. Moreover, process variation can also help hide HTs without hardware overhead. The functional analysis may be useful for HT detection at the RTL level. However, a stealthy HT can still evade detection using rare trigger conditions [20]. Overall, HTs can be carefully inserted with design optimisations by exploiting rarely activated/observed conditions and introducing ultra-low power/delay overhead to evade both post-silicon detection [60] and DFS techniques [64, 65].

Last, the emergence of new types of Trojan detection techniques will also pose challenges for Trojan design methods. Normally, HTs are stealthy with rare trigger events. As a result, HTs are usually not sensitised with test patterns during functional tests [66]. Therefore, researchers can focus on such rare events and hidden corners for Trojan detection. Hicks *et al.* [66] proposed such a method, known as unused circuit identification (UCI), which searches for unused components of an IC during design-time testing and marks them as potentially malicious. The UCI algorithm can detect many of the existing HTs reported in the literature, including most of the benchmarks in Trust-hub [49], which poses a new challenge for HT implementation.

Protection and prevention suggestions: The security challenges faced by the designers are obvious. Attackers at this stage have very high flexibility to implement any malicious function [17]. Furthermore, the earlier the Trojans are inserted, e.g. in the higher-level specifications or the RTL code, the harder for Trojan detection at later stages since it is impossible to obtain a golden RTL model [18].

From the designer's perspective, considering the motivations (goals) and capabilities of the design-stage attackers, the following countermeasures can be used against RTL Trojans: formal code checks (pre-silicon detection), SCA (post-silicon detection), and DFS techniques. Nahiyan *et al.* [67] proposed a technique to analyse and quantify the vulnerabilities in a finite state machine (FSM). The state transition table of a FSM, including do not-care transitions and states, is exacted from a gate-level netlist, and then used for vulnerability analysis. Xiao *et al.* [68] proposed a built-in self-authentication (BISA) method to prevent HT insertion during physical design, which uses functional cells to fill all the unused spaces in the circuit layout. As the unused spaces in a circuit are the most likely insertion area for Trojans, any component changes in the BISA structure could be detected. The BISA structure is vulnerable to several attacks, therefore, Shi *et al.* [69] proposed an obfuscated BISA structure to enhance its security. Other DFS methods include the ring oscillator (RO)-based technique [64, 65], split manufacturing [70, 71], and so on. As a supplementary method, runtime HT detection approaches, e.g. chaos theory based runtime power consumption monitoring [72], runtime data anomaly detection based on change point [22], have also been proposed.

Overhead: Formal code checks and SCA do not introduce overhead to the circuit. However, the DFS techniques usually add a new structure to the design thus will bring hardware overhead to the circuit. The power and area of the circuit will increase, and the performance of the circuit may be degraded due to the operations of the DFS structure.

Case studies: A summary of HT designs at the RTL level proposed in the literature is shown in Table 1.

(i) *Cyber Security Awareness Week (CSAW)*. Since 2008, the annual embedded system challenge (ESC) competition, which is held as a part of CSAW, is well-known for its HT competition. The competition targets HT design and insertion techniques, Trojan detection approaches, and design hardening mechanisms. Many researchers have reported their Trojan designs implemented for the CSAW ESC competition [16, 21, 25, 26, 73].

Several Trojan design and implementations have been presented by Baumgarten *et al.* [16] at CSAW ESC, including the following: information leakage through RS232 end sequence; RS232 multiple transmission rates; denial-of-service (DoS); thermal leakage; information leakage through amplitude modulation transmission; 50 MHz transmission; light-emitting device transmission. Jin *et al.* [21] presented eight RTL HTs to compromise the security of an Alpha encryption module. Santos and Fei [25] presented a backdoor Trojan, a bomb counter Trojan, and a power sink Trojan to weaken an 8051 processor performing RC-5 encryption. Reece *et al.* [26] presented DoS and data leakage Trojans to attack the Intel 8051 micro-controller unit, which would probably run a data-sensitive encryption algorithm. Karri *et al.* [27] presented several case studies of HTs from ESC, including the following categories: key/information leakage through video graphics array (VGA) display, the RS232 protocol, or temperature; synthesis tool-based Trojan; DoS Trojan.

In summary, these Trojans' payloads can be classified into three types as follows:

- leak sensitive data/internal signals;
- change the function of the design, or cause DoS;
- destroy the chip.

Also, the triggers of these Trojans can be divided into three types:

- input pattern triggered if the attacker can physically access the device;
- triggered internally by an internal event or sequence;
- always-on.

(ii) *Crypto-cores*. HTs can be carefully designed to compromise the security of widely used crypto-cores, which may be of particular interest to an attacker. In [23], a malicious circuit was developed to connect the encryption module and decryption module in an AES core. The Trojan is triggered when a predetermined condition is satisfied and then half-encoded data is sent from the encryption module to the decryption module by a specific Trojan path [23]. Therefore, plain text is directly sent to the output. Moreover, when a predefined keyword is inputted to the AES core, which is transferred to a controller through the Trojan path, the secret key is outputted directly [23].

In [32], key leakage HTs are demonstrated in a wireless cryptographic IC containing an AES module and an ultra-wideband (UWB) transmitter (TX) module. The impact of malicious components is carefully hidden below the side-channel margins.

The key is leaked through parameter modulation, e.g. frequency or amplitude of the wireless transmission [32]. The adversary is able to retrieve the 128-bit key leaked by a 128-bit ciphertext block through a transmission power waveform sent by the UWB TX.

Table 1 Overview of hardware Trojan designs targeted at the RTL level

Works	Benchmark	Trigger	Payload	Overhead	Detectability
Baumgarten *et al.* [16]	Alpha device (ESC2008)	always-on; internal	information leakage; DoS	power: 0.192%–1.026%	functional: unlikely; SCA: likely
Zhang and Xu [17]	OpenRisc	external; internal	N/A	N/A	functional: unlikely; SCA: likely; UCI: unlikely
Fern *et al.* [18]	bus protocols, ARM processor	always-on	leakage; unprivileged access	area: AXI4: 0.5%–2.1% Flip-flops (FF), 0.4%–3%(LUT); SoC: 0.9% (FF), 1.2% (LUT)	functional: unlikely; SCA: unlikely; Formal verification: likely
Subramani *et al.* [19]	802.11a/g transceiver	always-on	information leakage	0.5 dB–0.75 dB extra power	functional: unlikely; SCA: likely; formal verification: likely
Tsoutsos *et al.* [20]	DES, eXtended Tiny Encryption Algorithm (XTEA), Pseudo Random Number Generator (PRNG)	external	change functionality	N/A	functional: unlikely; SCA: likely; static analysis: unlikely
Jin *et al.* [21]	Alpha encryption	external; internal	leakage; compromise functions; destroy the chip	area: −9.4% to 3.3% (FF); 0.024%–6.8% (LUT)	functional: unlikely; SCA: likely
Kumaki *et al.* [23]	AES	predefined rule/input keyword	leakage	area: 0.37%; power: 0.13%	functional: unlikely; SCA: likely
King *et al.* [24]	Leon3 processor	a sequence of bytes; predetermined bootstrap	privilege escalation; login backdoor; stealing passwords	area: 0.075%	functional: unlikely; SCA: likely
Santos and Fei [25]	8051	external; internal; always-on	leakage; disables/ enables functions	area: 0.2%	functional: unlikely; SCA: likely
Reece *et al.* [26]	8051	external; internal	DoS; leakage	area: 0.15%–0.4%; leakage power: 0.146%–0.399%; dynamic power: −0.433% to 0.93%	functional: unlikely; SCA: likely
Karri *et al.* [27]	crypto-core etc. (ESC works)	external; internal; always-on	leak secret key/info. through the RS232 protocol, through temperature using SCA, or through VGA display; DoS	N/A	functional: unlikely; SCA: likely; UCI: likely
Dash *et al.* [28]	modern computers	a certain temperature	N/A	area: high; power: high	functional: unlikely; path delay-based: unlikely; power-based: likely
Gallais *et al.* [29]	RSA, AES	specific instructions; particular input	leak info./secret key	N/A	functional: unlikely; SCA: likely
Reece and Robinson [30]	AES	external; internal; always-on	leakage; drains the battery	area: 90 nm, 0.16%, 45 nm, 0.78%; leakage power: 90 nm, 0.53%, 45 nm, 0.46%; dynamic power: 90 nm, 2.59%, 45 nm, 0.49%	functional: unlikely; SCA: likely
Sturton *et al.* [31]	Leon3 processor	external	change functionality	N/A	functional: unlikely; formal analysis: likely; UCI: unlikely
Liu *et al.* [32]	wireless cryptographic IC	always-on	leakage	area: 0.005% and 0.025%; power: 0.4% and 0.1%	functional: unlikely; SCA: likely
Lin *et al.* [33]	crypto-processor	always-on	convey secret information off-chip	N/A	functional: unlikely; SCA: likely
Fyrbiak *et al.* [34]	AES	always-on; conditionally	cancel self-tests; key leakage	N/A	functional: unlikely; SCA: likely; formal verification: likely
Kaji *et al.* [36]	universal asynchronous receiver transmitter (UART)	always-on	facilitate data injection attack	N/A	functional: unlikely; SCA: likely

In [29], HTs have been proposed to introduce or amplify side-channel leakage of cryptographic software. Particularly, they implement several alterations to cause information leakage through faulty computations or the variations in the power consumption and latency of some instructions [29]. Software-based Trojan activation mechanisms are proposed and the side-channel leakage of Rivest–Shamir–Adleman (RSA) and AES implementation were illustrated [29]. Lin *et al.* [33] proposed an HT, which conveys secret information off-chip by employing power side-channels. By using a spread-spectrum technique, the information is leaked below the noise level of the AES circuit to evade detection. Each key bit is modulated by a simple XOR operation with a pseudo-random number sequence [33]. Fyrbiak *et al.* [34] proposed a framework to reverse engineer the gate-level netlists and insert HTs to weaken cryptographic circuits.

(iii) *Exploiting unspecified specifications, or creating covert channels*. Attackers can also carefully design HTs to hide in unspecified specifications or behaviours. Fern *et al.* [18] highlighted that current system-on-chip (SoC) bus implementations are vulnerable to HTs which can hide in the partially specified specifications or behaviours. They present a Trojan which introduces a covert channel by modifying bus signals of unspecified behaviours. The Trojan communication channel is demonstrated on a SoC design that runs a multi-user Linux operating system to allow an attacker to get root user's data without permissions [18]. It is shown that there are some redundant bus signals which will be ignored by the verification test, thus can be exploited for HT implementation [18]. Subramani *et al.* [19] demonstrated an HT in a wireless network by exploiting the forward error correction block to create a covert channel. Similarly, Kaji *et al.* [36] proposed a data injection attack by exploiting HT to create specific electromagnetic waves as a covert channel.

(iv) *Remote activation*. Since an attacker may have limited physical access to deployed devices, triggering HTs remotely is an ideal choice. Dash *et al.* [28] proposed a method to remotely activate HTs through a stealthy temperature channel. An analog HT trigger is implemented on modern computers, which can be remotely activated when the infected circuit reaches a predefined temperature [28]. The temperature of the target computer can be raised remotely by sending a large number of network requests to the computer.

(v) *Evading UCI detection.* To evade UCI detection, Sturton *et al.* [31] constructed malicious circuits that have hidden behaviours. Particularly, this class of malicious circuits satisfies the following property: for all signal pairs (s, t), there is at least one input that could make $s \neq t$ and would not trigger the hidden HT [31]. This property ensures that the UCI technique will not mark the circuitry between s and t as a potential HT. Exhaustive enumerations of all circuits satisfying that class are performed, and the search results are used to construct an attack on a processor, i.e. the Leon3 processor. This HT allows user-level programmes entering into supervisor mode to take control of the system by using a secret knock [31].

Zhang and Xu [17] proposed an HT design methodology from three aspects to bypass existing defenses, especially the UCI technique. First, to evade functional tests, carefully selected rare trigger conditions are used to make the HTs remain dormant during testing. Second, to evade UCI detection, they combine the trigger input selection and the code writing style to mask HTs as useful circuits. Third, they introduce a metric, namely un-controllability, to represent the difficulty level of setting the value of a signal [17].

4 3PIP vendor attacks

HT attack scenario: In this adversarial model, the 3PIP used by a design house or a SoC developer may contain HTs, as shown in Fig. 2. This is a general threat since SoCs are usually integrated with many 3PIPs with the purpose of reducing the cost and accelerating the time-to-market [1–3, 12]. HTs could be inserted at each type of the IP, e.g. soft for RTL-level, firm for netlist-level, hard for Graphic Database System II (GDSII) cores [15]. The SoC developer, who integrates design blocks and modules, often treats the 3PIP as black boxes. These unknown IP cores are usually unmodified and integrated into the final design, which can lead to an effective attack to compromise the SoC.

Motivation: Inserting a Trojan in 3PIPs is an effective and stealthy way for an attacker to compromise the security of SoCs. The attacker from a 3PIP vendor may want to insert an HT in the IP, which serves as a backdoor to steal secret information from the integrated SoC or cause functional failures of the SoC. The attacker can also insert an HT to facilitate a future attack. For example, implant a hardware backdoor to support the software or system attacks.

Feasibility: Untrusted 3PIP vendors can easily introduce stealthy malicious modifications to design through insertion, deletion, or modification of original circuits in a stealthy manner. This type of attacker can get the IP design files and source codes. Therefore, an attacker can flexibly implement malicious functions by modifying the IP design at RTL or other specification levels.

However, the 3PIP attackers also face some obstacles. First, it is difficult for the 3PIP attacker to physically access the fabricated ICs to trigger the HT. Therefore, the HTs are normally triggered internally. The HTs can also be designed to be always-on. Second, as the attackers insert HT in the 3PIP without knowing the overall design of the IC, it is not easy to carry out an attack successfully. In conclusion, the 3PIP Trojan has high feasibility, good practicality, and is easy to implement. The only limitation is the method of triggering.

Detectability: We now discuss the available detection techniques from the defender's perspective, and the anti-detection capability from the attacker's perspective. Pre-silicon detection methods, e.g. formal verification and code analysis, are usually utilised to detect HTs in 3PIP cores [74]. Previous research studies [62, 75, 76] have proposed hardware description language (HDL) code analysis or structural analysis techniques for soft IP cores. A SoC integrator can analyse the IP source codes and find potentially suspicious components by analysing the reachability and controllability [77]. However, the complexity of such an analysis method is extremely high, which increases with the circuit size exponentially [78]. In formal verification methods, IPs are verified by proof-checking tools to avoid including unintended functionalities [79–82]. Design-for-trust techniques have also been proposed. For example, Liu *et al.* [83] detected malicious HTs by applying security constraints to the task scheduling step of the SoC design process. Rajendran *et al.* [74] involved design constraints by using high-level synthesis to detect Trojans and then isolated the Trojan-infected 3PIPs. It is shown that using a variety of vendors can prevent collusion of multiple IPs from one vendor.

Fortunately for the attacker, verifying the trust of IP cores obtained from untrusted third-party entities is challenging due to several issues. For the case of 3PIPs, the methods depending on a golden chip/model are not suitable anymore. Moreover, the complete implementation of a 3PIP is invisible. It is difficult to provide sufficient coverage by general functional simulations due to incomplete functional specifications. It is also difficult to predefine comprehensive security rules to cover all the possible risks. There can always exist HTs, which can satisfy proof-checking constraints thus evading detection. Lastly, the HDL and Coq [35] (an interactive theorem prover/proof-assistant) representations of a circuit may not be completely equivalent. Even if the Coq representations of the circuit are verified to be trustworthy, it cannot guarantee that the corresponding HDL code is trustworthy [74]. A smart attacker can ensure that the functional specification of the design is unchanged or the modifications are undetectable.

Protection and prevention suggestions: Pre-silicon detection methods, *e.g.* formal verification methods, can be used to detect HTs that are inserted in 3PIP cores, while post-silicon detection methods usually cannot detect 3PIP Trojans. The DFS techniques, such as using a variety of vendors, can also be used to prevent 3PIP Trojans.

Overhead: The formal verification methods will not introduce overhead to the circuit, while the DFS techniques usually bring some overhead to the design.

Case studies: Tsoutsos *et al.* [20] presented HTs which do not violate the functional specifications. Multiple levels of malicious nested FSMs are introduced to the design. The threat scenario is that the SoC integrator, which receives the malicious IP only applies static analysis methods on the HDL code of the IP, without actually simulating or implementing the design (dynamic analysis) [20]. Such modifications are hard to detect without exhaustive testing of all system states.

While HTs are generally considered to be malicious, they may also go in the opposite direction, e.g. be exploited in a constructive way. In [84], a hardware IP protection technique that embeds an HT as a FSM was proposed. By using a sequential Trojan that acts like a time-bomb, illegal SoCs containing pirated evaluation copies of the IP could stop performing the specified functionality. Specifically, on the occurrence of a rare sequence, the Trojan stops the normal usage of the IP [84]. Therefore, expiry date on the usage of the IP can be set based on the Trojan.

So far, most research works consider side-channels as undesired signals such that people need to protect devices from sophisticated SCA attacks. However, Gallais *et al.* [29] used side-channel leakage introduced by an HT as a watermark for IP protection, which can be detected by SCA. A unique signal is embedded into the side-channel signal of a circuit which acts as a watermark. This enables circuit designers to detect unauthorised use of their circuits. They illustrate this by designing an integer-based multiplier [29]. When a specific pair of operands arrive, the pipeline will be stalled for several clock cycles. Since this pair of inputs is hard to guess, it allows the designer to verify his own IP by analysing the power profile [29].

5 CAD tools attacks

HT attack scenario: Untrusted commercial CAD tools supplied by different vendors can also introduce malicious circuits to a design, which reflects the synthesis and verification stage attacks, as shown in Fig. 2. CAD tools attackers can directly insert Trojan circuits into the design or degrade critical logic, e.g. the random number generator (RNG) used in a cryptographic circuit.

Motivation: The attacker from the CAD tools providers may want to insert HTs in the design files, which leaves an undetectable backdoor or a time bomb in these designs. The attacker can also control these systems after deployment or steal secret information from those systems.

Feasibility: Although this attack model is less possible compared with design attacks and fabrication attacks, it is still feasible. A CAD tools attack is more powerful and stealthy than design attacks and fabrication attacks. A SoC designer has to design chips by relying on CAD vendors. By compromising the CAD tools or the running scripts, the attacker can introduce a malicious modification to IPs from the HDL codes to the generated netlist [16]. In conclusion, the CAD tool attacks are feasible, having good practicality and stealthiness, but are not easy to implement.

Detectability: Since this attack happens during the synthesis stage on generally trusted tool suits, it is not suspicious and extremely hard to detect [16]. On the other hand, Trojans inserted by CAD tools are difficult to detect or remove since they are coupled with other design units [85]. Furthermore, a SoC designer often uses a suite of CAD tools supplied from the same vendor, which means the verification tool could potentially hide the malicious alterations introduced by the synthesis tool from the same vendor [1, 16].

Protection and prevention suggestions: A trust evaluation of CAD creators and security policies, which are currently lacking, need to be established to defeat the malicious tampering by CAD tools [16]. It is suggested to use reliable CAD tools or use self-developed CAD tools. Pre-silicon and post-silicon Trojan detection techniques are also needed to be applied.

Overhead: The cost of developing CAD tools independently is high. However, there will be security threats when using third-party CAD tools.

Case studies: Pilato *et al.* [86] demonstrated the CAD threats by compromising a high-level synthesis tool to insert three HTs. The payloads of these Trojans are adding latency, compromising the security of crypto-cores, and draining energy, respectively. Similarly, Pilato *et al.* [85] use high-level synthesis to inject a benign HT, which serves as an IP watermark to prevent piracy and counterfeiting. Basu *et al.* [87] investigated the CAD attacks from all the CAD tools (from synthesis, design, verification to test), and show that all these CAD tools can launch potential attacks. They demonstrate the CAD-attacks on an ARM Cortex processor.

6 Fabrication stage attacks

HT attack scenario: This attack model represents the threat of untrusted foundries. Nowadays, most modern ICs are manufactured worldwide in untrusted foundries due to budget considerations. The foundry receives the complete design (physical layout geometry file) and its specifications. However, the IC designer has little or no control over the foundries. A fabrication attacker could directly insert an HT into the chip or changing the manufacturing process to cause reliability issues in the SoCs.

Motivation: A fabrication stage attacker may want to cause reliability issues in the SoCs, steal information from the ICs, or even directly destroy the system.

Table 2 Summary of HT design and implementation works at the layout level

Paper	Benchmark	Trigger	Payload	Overhead	Detectability
Becker *et al.* [7]	RNG, AES	always-on	change functionality; degrade performance; leakage	area: 0	functional: unlikely; optical inspection: unlikely
Ghandali *et al.* [6]	32-bit multiplier, ECDH key agreement protocols	violating the delays of rare combinational logic paths	Trojan multiplier computes faulty outputs	N/A	functional: unlikely; visual inspection: difficult; SCA: difficult
Wang *et al.* [73]	ESC2010	rare events	change function	area: 0.6%; power: 0.4%	functional: unlikely; SCA: unlikely
Bhasin *et al.* [37]	cryptographic IP	external; internal	facilitate DFA; Leakage	area: 0.5% (LUT)	optical imaging: likely
Yang *et al.* [9]	OR1200	internal	privilege escalation; change functionality	area: 0.08%; delay: 0.33%	functional: unlikely; SCA: unlikely
Kumar *et al.* [38]	PRINCE	slightly reduced supply voltage	facilitate attacks	area: 0	functional: unlikely; SCA: likely; optical inspection: unlikely
Shiyanovskii *et al.* [39]	static random access memory	always-on	reduce the reliability by the acceleration of the wearing out mechanisms	N/A	functional: unlikely; delay monitoring: difficult; RO: difficult; wafer and package level reliability monitoring: likely
Lin *et al.* [40]	Crypto core	always-on	convey secret information	area: 14 LUTs	functional: unlikely; SCA: likely

Feasibility: The foundries have complete access to the layout of the design, which provides them with opportunities to flexibly add or remove components of ICs by modifying the layout. Since the foundry has no access to the RTL code, the modifications can only be achieved in the layout by changing the netlist, or modify the manufacturing process by changing design masks to not affect the functions of the design [7], as shown in Fig. 2.

Generally, it is not easy for an attacker to insert HTs during fabrication. First, the attacker has to figure out the exact functionality of the circuit (in the form of the GDSII file). The attacker also needs to find the necessary space to add extra gates and connections. Second, the attacker needs to keep the layout (place and route) almost identical, to avoid being detected by optical inspection. In conclusion, the fabrication stage attacks are feasible, having good practicality and stealthiness, but are not easy to implement.

Detectability: A Trojan inserted during fabrication is difficult to discover by functional tests and verification performed on the HDL. When the layout of the circuit remains unchanged during Trojan insertion, it is almost impossible to detect these Trojans by using optical inspection. An attacker can insert HTs based on the modification of the electrical characteristics while the metal, active area and poly-silicon layer remain unchanged [7].

Fabricating a golden chip in a trusted factory for HT detection is difficult in practice. Thus, the detection technique can only compare the golden simulated model and the fabricated chip under test. However, Yang *et al.* [9] show that an analog HT can be much smaller and more stealthy than a digital HT. The trigger is implemented by diverting charge from unlikely signal transitions, which makes the Trojan invisible to side-channel detection.

Protection and prevention suggestions: Optical inspection is considered as a reliable way to detect layout-level HTs, while SCA is also a general method to detect this kind of HT. Since the RE-based optical inspection needs a lot of human efforts, it is also helpful to use a machine learning-based image analysis method for automatic analysis [88]. Besides, a golden simulated model, if exists, will be helpful for post-silicon detection techniques [89], e.g. SCA.

Overhead: Post-silicon detection does not introduce overhead to the circuit, while the optical inspection will consume a lot of human labour and time costs.

Case studies: As most of the HTs reported to date in the literature are inserted at the RTL level, constructing practical HTs at the layout level is still an open problem. The summary of HT design and implementation works targeting at the layout level is shown in Table 2, which will be discussed in the following paragraphs.

(i) *Exploit analog circuits:* It has been shown that an attacker during fabrication can exploit analog circuits to create small and stealthy HTs [9]. Yang *et al.* [9] leveraged analog circuits to perform a hardware attack, named A2. A circuit is constructed using capacitors to siphon charge from nearby wires in the spare spaces of a design after place and route. A victim flip-flop is changed to the desired value when the capacitors are fully charged. This attack has been implemented in an OR1200 processor by applying it to privilege escalation which can be controlled remotely [9].

(ii) *Parametric Trojans:* Becker *et al.* [7] proposed layout-level Trojans by slightly altering the manufacturing process conditions, i.e. the dopant polarities of a transistor. The Trojan can accelerate the wear-out mechanisms to affect the reliability of ICs. The Trojans have been inserted into two designs, i.e. an Intel secure RNG in an Ivy Bridge processor, and a side-channel attack resistant substitution box (S-Box) implementation [7].

Ghandali *et al.* [6] presented a parametric Trojan, which is designed through modifying the parameters of transistors, and does not require extra logic. It is triggered under rare conditions that are determined by the delays of some combinational logic paths. This design has been applied in a multiplier circuit to create a Trojan multiplier. If specific patterns are input, this Trojan multiplier will compute faulty outputs [6]. This Trojan multiplier is further applied to attack a key agreement protocol, the elliptic curve Diffie–Hellman (ECDH). The bug attack works as follows [6]. First, the first several bits of the key are guessed. Then, a point Q which can lead to a failure of the scalar multiplication is searched. After that, the attacker sends Q to the server to make a handshake, which performs the ECDH protocol. If the handshake fails, it indicates that the Trojan multiplier outputs the expected multiplication error. Hence, the current guessed key is correct. More bits will be cracked successively in this way to recover the key [6].

Kumar *et al.* [38] proposed parametric manufacturing process HTs to facilitate fault-injection attacks. The Trojans are inserted by altering the doping concentration and the dopant area of predetermined transistors in a target circuit. The trigger condition of the HTs is slightly reduced supply voltage with very low probability [38]. The Trojans have been utilised to inject faults into the lightweight cipher PRINCE. It is shown that they can reconstruct the secret key after around five fault-injections by differential cryptanalysis [38].

Shiyanovskii *et al.* [39] proposed an HT based on process reliability. The Trojan reduces the reliability of ICs by altering the conditions of the manufacturing process, to wear out complementary metal oxide semiconductor transistors. Such minor changes in the manufacturing process are hard to detect.

(iii) *Unchanged place and route*: Bhasin *et al.* [37] analysed how to introduce an HT while the place and route remain unchanged. It is shown that when the placement density is over 80%, it is difficult to insert Trojans. They also inserted a Trojan to aid the differential fault analysis (DFA) attack. The payload of the HT is an XOR gate that alters one bit of the AES to be faulty in the eighth round. As a result, the attacker can retrieve the whole key by activating the HT for two encryption processes [37].

Wang *et al.* [73] considered new placement techniques and delay-aware Trojan insertion. A hard macro is used to prevent delay variations in FPGAs. For the application specific integrated circuit (ASIC) scenario, where the Trojan is inserted at post-layout, the placement and route of the original design are also preserved by making it a hard macro [73]. It is shown that such Trojans only have a small impact on path delay, which can evade on-chip monitor-based DFS approaches.

(iv) *Trojan side-channels (TSCs)*: Lin *et al.* [40] used side-channel leakage for HT implementations, called TSCs. A hidden backdoor can be inserted at the foundry for unauthorised leakage of secret information. Power side-channels are demonstrated to leak information that can be hidden in the noise. Two TSCs are implemented, i.e. TSC based on spread-spectrum theory and TSC using specific input values [40]. Moreover, the TSCs have physical encryption property, so that it can keep the information secure even if the introduced side-channel is successfully detected.

Table 3 presents a comparison between Trojan insertion at the RTL level and layout level from the attacker's perspective. The advantages of Trojan insertion at the RTL level are having full

Table 3 Comparison of Trojan insertion at the RTL level and layout level from the attacker's perspective

	RTL level	Layout level
Pros	1) have full access to the source code	1) can evade detection by functional testing and verification
	2) high flexibility to implement any malicious function	2) may be invisible to SCA
	3) since an RTL modification will affect all the fabricated ICs, a golden model may not exist for SCA	3) can leverage analog circuits or parameter changes to introduce small and stealthy Trojans
Cons	1) can be revealed by complete code reviews and adequate security policy checks	1) have to make modifications to the layout mask or at process level which is neither easy nor flexible
	2) may be exposed by SCA	2) must keep the original layout mostly unchanged to avoid being detected by optical inspection

access to the source code and high flexibility to implement any malicious function. Moreover, as an RTL modification will affect all fabricated ICs, a golden model may not exist for SCA-based detection methods. The disadvantages of RTL Trojans are that they can be exposed by complete code reviews, adequate security policy checks, or SCA. On the other hand, the advantages of Trojan insertion at the layout level are that it can evade detection by functional testing and be invisible to side-channel defenses. It can also leverage analog circuits or parameter changes to introduce small and stealthy HTs. The disadvantages of Trojan insertion at the layout level are that it is not easy for the attacker to make modifications to the layout mask or change the manufacturing process. They must also keep the original place and route mostly unchanged, to avoid being detected by optical inspection.

7 Testing stage attacks

HT attack scenario: In general, the manufacturing test is done by a credible test party, e.g. reputable semiconductor company or government agency, which could be considered as trusted. As a special case, Xue *et al.* [41] formulated untrustworthy testing parties into two attack models and illustrate that the test parties may be untrustworthy. In [42], Yasin *et al.* extract secret information from test data. These attacks indicate that the testing phase may also be insecure. The testing party is important in the IC supply chain. However, nowadays, there is usually only one test party to test the fabricated ICs. If the only testing house is not credible or colludes with attackers from other stages [90], the testing results will no longer be trustworthy.

Motivation: An attacker during the test stage may want to modify the HT detection results or modify the test data to hide the HTs.

Feasibility: Generally, the testing party collects test data of fabricated ICs and then performs the HT detection procedure. In this scenario, the testing agency can directly modify the Trojan detection results. In a special case, the designer is involved in the testing process. In this scenario, the test agency needs to modify the test data to mislead the final Trojan detection result. An adversarial test data generation algorithm was proposed in [41] for the untrustworthy testing houses, which can use the minimum test data modifications to cause the maximum detection errors of ICs. In conclusion, the testing stage attacks are feasible, having good practicality, but are not easy to implement, as shown in Fig. 2.

Detectability: Little research has been done on testing stage defenses. Xue *et al.* [41] proposed an HT detection method based on hybrid clustering ensemble to resist untrustworthy testing houses. Three testing houses are used in the scheme, and each testing house carries out the HT detection process. Then, the three detection results are consolidated by using the hybrid clustering ensemble method to obtain the final test result. The technique can resist malicious modifications by untrustworthy testing houses, and can achieve higher detection accuracy than each of the three testing houses regardless of whether the testing house has maliciously modified the test data or not [41].

Protection and prevention suggestions: Since the motivations of the testing stage attackers are to modify the HT detection results or modify the test data to hide HTs, two methods can be applied to resist such attacks. One is the trust evaluation of testing parties, and the other is the ensemble technique using multiple testing parties [41].

On the other hand, there have already been a few secure testing methods against IC piracy, which may provide a reference for secure testing of HT detection. For example, Contreras *et al.* [91] present a secure split-test (SST) technique to prevent counterfeiting. The method allows the IP owner to meter the IPs by holding a lock key. During the test phase, a key is required to unlock the IP's functionality, so that the IP owner can verify the testing results. Later, Rahman *et al.* [92] improve the above SST technique against IP piracy by simplifying the communication between the IP owner and the foundry, named CSST. In the CSST method, the IP owner controls the testing by locking the IC and the scan chains [92]. Only the IP owner can understand the testing results under locking conditions, and can unlock the IC. Zhang *et al.* [93] proposed a hybrid approach that combines a dynamically obfuscated wrapper technique (referred to as DOST) and SST to protect IP rights, which allows the IC designer to control the fabrication and the testing processes. In the locked model, structural tests are performed, while in the unlock model, the functional tests can be performed [93].

Overhead: Since multiple testing houses are used, the cost will be of particular concern. It is shown in [41] that the time overhead of ensemble technique is small and acceptable, while the computational overhead is large. However, the computational overhead is distributed across multiple testing parties, which means that the ensemble technique does not increase the computational and storage overhead of each test party.

Case studies: To date, little research [41, 42] has been done on testing stage HT attacks, as described above.

8 Distribution stage attacks

As described in Section 2, since the distributor is usually unaware of the IC design, it is generally considered that the distributor cannot insert an HT. However, a distribution stage attacker can RE a chip to pirate the chip, or directly replace the IC with a Trojan-inserted version during transportation. Therefore, we also describe the attackers from the distribution stage.

HT attack scenario: After IC fabrication and packaging, a distribution attacker may appear in the IC supply chain. Such a distribution attacker, which may be either an IC distributor or a user, is restricted in inserting Trojans. Instead of being able to modify logic gates, they have to destroy the package or the components, or manipulate the transport process [16], as shown in Fig. 2.

Motivation: A distribution stage attacker may want to RE a chip to pirate the chip. The attacker may also want to directly replace the circuit with a Trojan-inserted circuit during transportation.

Feasibility: An attacker has limited flexibility at this stage and it is difficult to implement such hardware attacks. Such attackers cannot obtain the HDL code and layout level geometry. The attacker also does not have the input/output test patterns. However, they usually have a set of specifications about the functions of the ICs. They may obtain the netlist of the design by RE, which is a difficult but feasible task. In conclusion, the distribution stage attacks have limited flexibility, some practicality, and are difficult to implement.

Detectability: Some defense techniques have been proposed to address this type of vulnerability, including anti-tampering packaging and obfuscation against SCA [16]. HT attacks and defense techniques at this stage remain open problems.

Protection and prevention suggestions: Since the motivation of a distribution stage attacker is to RE or replace the circuit, logic obfuscation and logic encryption can be used against RE attacks. Some fragile hardware watermarking structures can also be used. Once the integrity of the hardware is compromised, the watermark will be broken.

Overhead: The DFS techniques will add hardware overhead to the circuit.

Case studies: Swierczynski *et al.* [43] described a HT attack on a USB flash drive. The USB flash drive is intercepted and attacked during transportation. The FPGA bitstream is manipulated such that the S-Box of the 256-bit AES design is changed to a linear function, and thus can be easily broken [43]. If the attacked USB flash drive is used by a victim, the user's data can be revealed from the ciphertexts.

9 FPGA Trojans

HT attack scenario: With the extensive use of FPGAs in critical applications, the security of FPGA designs has become a major concern. In the past, the research studies have focused on IP protection in FPGA, i.e. protecting the IP mapped on an FPGA from being stolen. However, little research has been conducted on the security and protection of the FPGA device itself. Recently, a few FPGA HT detection techniques have been proposed, while the FPGA-oriented HT design and implementation works are relatively less.

Motivation: The attacker may want to cause malfunction of the FPGA system, steal secret information, or lead to other unauthorised operations.

Feasibility: Similar to the ASIC scenario, malicious modifications of the FPGAs are possible at several stages of the design and fabrication process. An attacker can create an FPGA Trojan by directly modifying the HDL, modifying FPGA fabric, modifying physical parameters, modifying the bitstreams, or by exploiting FPGA CAD tools [94]. For example, a malicious circuit can be inserted by an adversary to monitor the logic values of internal nodes, logic modules, and the look-up tables (LUTs) [95]. Once the Trojan is triggered, the FPGA can malfunction in different ways, e.g. the LUT values can be changed, configuration cells can be altered to perform incorrect routing, or incorrect values can be written into block-random access memory [95]. In conclusion, the FPGA Trojan attacks are feasible, having good practicality, but are not easy to implement, as shown in Fig. 2.

Detectability: These Trojans can escape common FPGA testing that cannot cover all possible triggering conditions. Existing FPGA Trojan defense techniques fall into two categories, SCA and DFS techniques. The power consumption based [47] and electromagnetic emanation (EM) based [96] SCA methods are proposed to detect FPGA HTs. Chen *et al.* [97] measured the EM of the FPGA clock tree, and use principal component analysis for signal processing. Then, backpropagation neural network is used to automatically detect the FPGA HTs. Similar to fingerprint-based HT detection methods in ASIC scenarios, FPGA detection methods based on anomaly features have also been proposed. Pino *et al.* [98] proposed a process variation-based anomaly detection method for FPGAs, which can isolate suspicious Trojan areas with inconsistent characteristics. In their later work [99], after isolating these suspicious areas, the remaining trustworthy areas, named FPGA Trust Zone, are selected to run the designs securely.

On the other hand, some DFS techniques are also proposed against FPGA Trojans. Mal-Sarkar *et al.* [95] proposed a redundancy-based approach using Trojan tolerance, which modifies the application mapping process to provide defenses against HTs. Swierczynski *et al.* [100] used the dynamic obfuscation of cryptographic primitives to prevent the bitstream RE and modification-based FPGA HTs. Bloom *et al.* [101] proposed a FPGA HT defense technique, named MORPH, which uses onion-encryption for encrypted execution and uses a specific hardware abstraction layer to isolate the hardware and software. Zhang *et al.* [102] used the moving target defense principle to prevent FPGA CAD tools based on Trojan insertion, in which three kinds of unpredictability are introduced into the FPGA designs.

Protection and prevention suggestions: Considering the diverse motivations (goals) and strong capabilities of the FPGA Trojan attackers, the defense against FPGA Trojans is still an open problem. SCA method can be used with the help of a golden model or built-in consistency verification structures. DFS methods, e.g. redundancy based mapping approach, dynamic reconfiguration, obfuscation, and structural isolation, are also promising protection methods against FPGA Trojans.

Overhead: The DFS techniques will bring some hardware overhead in terms of logic resources (area), power, and performance.

Case studies: The summary of FPGA-oriented HT design and implementation works is shown in Table 4. The HT type is based on the FPGA HT taxonomy proposed in [94]. Note that the FPGA Trojans implemented by using direct HDL modification are not included in this table, because those HTs are not specifically for the FPGA, but just using the FPGA device as a code verification platform.

Chakraborty *et al.* [44] inserted HTs into FPGA by directly modifying the unencrypted bitstream file. They implement a number of ROs as the HT in a 128-bit AES circuit, which can cause the temperature to increase thus lead to accelerated aging. Since this Trojan is inserted during the bitstream configuration, it does not leave traces in the logic and place and route phases [44]. Swierczynski *et al.* [45] proposed an FPGA bitstream Trojan implementation scheme, which detects the S-boxes in bitstreams, and then modifies the bitstream of S-boxes to weaken the AES and 3-data encryption standard (3-DES) algorithms. As mentioned in Section 8, Swierczynski *et al.* [43] proposed an interdiction-based FPGA Trojan insertion, which modifies the bitstream to replace AES S-boxes. They demonstrate their work on XTS-AES on Kingston DataTraveler 5000 to recover the plaintext. Krieg *et al.* [46] proposed an FPGA CAD tool Trojan, including malicious insertion during synthesis, and malicious part activation during bitstream generation. They evaluate the scheme using iCE40 design flow running an instruction decoder of a central processing unit (CPU) to launch a privilege escalation attack. Marchand and Francq [47] designed, placed, and routed 12 functional FPGA Trojans by hand on 128-bit AES on the SASEBO-GII Board (Xilinx Virtex-5), which can lead to DoS, changing specifications, or information leakage. Krieg *et al.* [103] implemented a Trojan trigger by exploiting the X-Optimism operations (on unknown 'X') in an FPGA simulation model. They generated a trigger signal, which is '0' during simulation phase and '1' in implemented hardware. FPGA HTs can also be used with a benign purpose. Zheng *et al.* [48] propose a functional Trojan to disable particular general-purpose registers, which works as a security mechanism for FPGA systems. They implement delay-logic arbiters as HTs and evaluate on OpenRISC OR1200 on Xilinx Spartan-6. These efforts demonstrate the flexibility of FPGA HTs.

10 Future directions

In this section, we will discuss the potential future HT implementation and detection techniques.

Table 4 Summary of FPGA-oriented HT design and implementation works

Works	Benchmark	HT type	Insertion mechanism	Trigger	Payload
Chakraborty *et al.* [44]	128-bit AES on Xilinx Virtex-II	bitstream Trojan	bitstream modification to implement many ROs as the HT	always-on	temperature increases thus accelerating aging
Swierczynski *et al.* [45]	AES and 3-DES	bitstream Trojan	detect S-boxes in bitstreams, then modify the bitstream of S-boxes	always-on	weaken the cryptographic algorithm
Swierczynski *et al.* [43]	XTS-AES on Kingston DataTraveler 5000	bitstream Trojan	bitstream modification replacing AES S-boxes during interdiction	always-on	recovering plaintext
Krieg *et al.* [46]	iCE40 design flow running an instruction decoder of a CPU	CAD tool Trojan	malicious insertion during synthesis, then activate malicious part during bitstream generation	output of the malicious LUT	privilege escalation
Marchand and Francq [47]	128-bit AES on SASEBO-GII Board (Xilinx Virtex-5)	functional Trojan	design, place and route the 12 HTs by hand	time-based, user, internal state	DoS, changing specifications, information leakage
Zheng *et al.* [48]	OpenRISC OR1200 on Xilinx Spartan-6	functional Trojan	implementing delay-logic arbiters as HTs	digital value	disable general purpose registers
Krieg *et al.* [103]	Xilinx 7	functional Trojan	exploiting the X-Optimism operations in an FPGA simulation model	always-on	the signal which is '0' during simulation becomes '1' in hardware

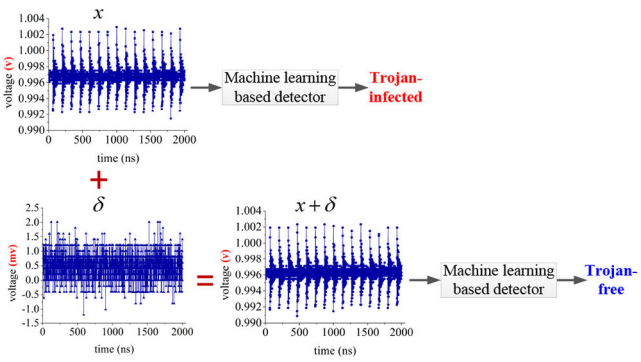

Fig. 3 *Illustration of adversarial HT designs against machine learning-based Trojan detection method: the original power trace x is detected as Trojan-infected. After introducing an imperceptible perturbation δ to x, the power trace x + δ will be detected as Trojan-free*

10.1 HT benchmarks and evaluation methods

A common concern is whether a real HT has been found in the industry. Owing to the sensitive nature of industry IP, it is unlikely that such HTs will be reported publicly. As such, standard benchmarks to evaluate HT implementations and defenses are highly needed. The Trust-HUB benchmark [49] developed by Tehranipoor *et al.* is well-known for its hardware security-related benchmarks. Trust-hub [49] currently provides the largest database of HT benchmarks and has been widely used in the literature. For example, Reece and Robinson [30] evaluated 18 AES HTs supplied from the Trust-hub database, in terms of power and area. It was shown that when spending enough effort into optimising the HT, the introduced overhead could be very small.

Furthermore, to create dynamic Trojan benchmarks, Cruz *et al.* [104] proposed an automatic HT insertion framework, which can insert HTs with validated trigger conditions and payloads in gate-level designs. It allows configurations, e.g. the type of the Trojan, Trojan trigger probability, and choices of payload [104]. Although this powerful Trojan automation design and implantation tool has emerged, Trojan design and detection is still a game process. Once the defenders know how these automatic tools generate HTs, Trojans inserted by these tools may also be easily detected. However, defenses always lag behind attacks. On the other hand, various new HT detection techniques have also been proposed. When attackers are aware of these detection methods, more powerful Trojan design methods will also appear.

To date, most of the Trojan implementation methods or Trojan detection techniques are verified under specific experimental conditions, specific stages and specific scenarios, and targeting specific circuits or Trojans. This non-uniform paradigm raises a question: which attacks (defenses) are more effective (universal)? To this end, a uniform evaluation method with comprehensive evaluation metrics is required to evaluate and analyse various HT implementation methods and defense techniques. Such a uniform evaluation method can ensure researchers and IC designers to: (i) evaluate the effectiveness of different HT attack and defense methods; (ii) assess ICs' vulnerabilities; (iii) carry out complete and quantitative comparative research studies on HT implementations and detection methods.

10.2 Machine learning-based Trojan detection methods, and HTs targeting machine learning models

Recent research in this field has explored machine learning methods for HT detection [41, 88–90, 105–111]. Generally, machine learning methods can be utilised for HT detection in the following aspects: providing automatic layout identification in RE-based methods [88, 105, 106], providing run-time HT detection architectures, which are trained by HT attack behaviours [107, 108], providing automatic feature analysis [112], and providing golden chips-free HT detection techniques based on classification or clustering [41, 89, 90, 109–111]. In particular, the machine learning method has its own specialties in feature extraction and image recognition, which makes it possible to reveal unknown HTs by monitoring suspicious behaviours and features. It can also improve detection capabilities through self-learning. Elnaggar and Chakrabarty [113] reviewed the works applying machine learning methods for hardware security. Specifically, they summarised that the machine learning methods can be used to classify or cluster the IC's parameters, gate-level nets, or traffic data in multi-core systems, for HT detection [113].

To defeat machine learning-based Trojan detection methods, attackers may introduce adversarial HT designs, which can make the detection methods produce incorrect decisions. In machine learning systems, adversarial input perturbations carefully crafted at the test stage can subvert the model's predictions on the instances. Attackers can investigate the vulnerability of machine learning models to such adversarial inputs (also known as adversarial examples) to mislead the HT detection. Xue *et al.* [41] proposed a data modification algorithm for untrusted testers to slightly modify the collected test data, to mislead the HT detection results. Such an example is illustrated in Fig. 3, in which the original power trace x of an IC is detected as Trojan-infected by the machine learning-based HT detection method [41]. After introducing an imperceptible adversarial perturbation δ to the test data, the power trace $x + \delta$ will be recognised as Trojan-free by the machine learning model [41].

On the contrary, there are also few works to study HT attacks targeting machine learning models and artificial intelligence chips. Clements and Lao [114] proposed to insert HTs in the functional block of the neural network implementations. As a result, the desired misclassification can be achieved when a specific input trigger arrives. Ye *et al.* [115] inserted an HT into the FPGA convolutional neural network (CNN) accelerator to launch an attack on a CNN-based image classification task. The HT can control the classification result once triggered. Odetola *et al.* [116] proposed an HT attack on deep learning models without modifying the parameters or functions within the layer. They exploit the statistical properties of each layer's output to trigger the HT, which makes the HT extremely stealthy. Li *et al.* [117] proposed a more flexible attack framework on a neural network that combines the hardware and software. Particularly, in addition to the hardware HT circuit, Trojan weights are embedded in neural networks. The Trojan is only inserted in a part of the network and does not affect the overall accuracy, thus can ensure stealthy [117]. In the above attacks, the attacker needs to have the knowledge of the model. Hu *et al.* [118] proposed memory Trojan on deep neural networks, in which the Trojan logic is only inserted into the memory controller without the knowledge of the model. Targeted attacks or untargeted attacks can be achieved when the trigger image arrives.

10.3 Attacks and defenses from chips to complex systems

Most of the existing Trojan attacks and detection are aimed at the chip level. A more practical scenario in the industry is how to implant and detect HTs on a complex SoC or larger systems. Such a system contains many components and connections. It also contains hardware, firmware, and software. This makes HT attacks more diverse, such as hardware-promoted software attacks, or software-promoted hardware attacks, or covert-channel attacks, and so on. It is important but challenging to detect HTs in such a complex system.

10.4 Universal Trojan and automatic Trojan insertion versus automatic Trojan (IC vulnerability) analysis tools

Most HTs reported to date are manually inserted into a specific target circuit [119]. However, a more ideal situation is that arbitrary Trojan circuits with arbitrary components could be inserted into any circuits, as shown in Fig. 4. There are two requirements for such practical attacks: (i) designing a universal Trojan independent of the host circuit, which is applicable for any given circuit; (ii) developing the automatic Trojan design and insertion tools. To meet these requirements, the automatic trigger and payload identification of a design at different levels are required.

Another type of universal Trojan is a general malicious hardware that can support a wide range of general-purpose attacks.

This is a more aggressive attempt. King *et al.* [24] presented two such hardware designs in Illinois Malicious Processors and exhibited three attacks using this hardware. Through the memory access mechanism, a privilege escalation attack was implemented, which gives the attacker root access without identification or creating system logs. Under a shadow mode, a login backdoor is implemented, giving an attacker authority to log in as a root user with no password needed [24]. Another function that steals passwords is also implemented.

In contrast, automatic Trojan analysis tools and automatic IC vulnerability analysis tools [120] are required. Most of the existing Trojan detection techniques are manually customised detection methods/scripts applied for specific scenarios and specific stages. The detection process requires manual participation, and the universality is limited. It is necessary to develop mature universal tools, including automatic Trojan detection tools and automatic circuit vulnerability analysis tools, to promote DFS and Trojan detection works.

10.5 Multi-stage HT attacks and defenses

The majority of previously reported HTs in the literature are inserted at a single stage in the IC's life cycle. However, malicious conspiracy between multiple entities at different stages in the IC supply chain could make HT attacks more powerful. Ali *et al.* [121] described such an attack on an AES implementation. They show that such a multi-stage attack is significantly stronger than an HT attack by a single entity, both in the life cycle of ASICs and FPGAs. As a result, it would be very difficult for current defense approaches targeting individual stages to detect such a distributed attack [121]. Detecting such a multi-stage Trojan is still an open problem. It is necessary to study universal Trojan detection methods independent of the stages.

10.6 Split manufacturing

Split manufacturing is a promising hardware security solution in the manufacturing stage where the untrusted foundries only know part of the design information thus makes it difficult for them to insert HTs. In recent works, different split manufacturing methods are proposed, e.g. [71], or combined with other hardware security techniques, e.g. layout camouflaging [70].

10.7 Low overhead runtime HT monitoring techniques

A large number of Trojan detection techniques have been proposed, but it is still possible for Trojans to escape detection and activate when the chip is used in the field. Runtime HT monitoring technique (which is relatively less in existing works) is a necessary supplement to Trojan detection and is also the last line of defense. However, existing runtime HT detection techniques suffer from high additional hardware overhead or high computational complexity. A low overhead runtime HT monitoring technique is a promising research direction, e.g. [22, 72].

10.8 Logic obfuscation for HT prevention

Logic obfuscation (logic encryption, logic locking) is a widely studied hardware security technique, which is usually used to prevent IC piracy and IC overbuilding. It can also be used as a DFS method to prevent HT insertion. Chakraborty and Bhunia [122] proposed a key-based obfuscation scheme to prevent HT attacks, in which two functional modes are introduced, i.e. obfuscated and normal modes. A large number of states have also been added to the obfuscated mode for obfuscation. This method prevents the attacker from finding the real rare states in the circuit [122]. Dupuis *et al.* [123] proposed a logic encryption approach to prevent HT insertion by minimising the number of rare events in a circuit. Similarly, Rathor *et al.* [124] proposed a logic encryption method using key-gate topologies to remove rare-triggered nets to thwart HT. Frey and Yu [125] proposed an approach using state obfuscation for HT detection. Illegal states caused by wrong keys are examined to detect HTs. They indicate that an attacker without the correct key cannot successfully modify the design without being noticed [125]. Yu *et al.* [126] reviewed the works on logic obfuscation for HT prevention and detection. They indicate that logic obfuscation can make it difficult for attackers to understand or RE the design thus can hinder the implantation of Trojans or can facilitate the HT detection after manufacturing. Similar to ASICs, obfuscation can also be used to protect FPGA designs. Hoque *et al.* [127] proposed an obfuscation-based approach against bitstream modification attacks on FPGAs. Particularly, the critical functions in an FPGA design are identified and masked (obfuscated). Besides, they use a redundancy technique for obfuscation to thwart tampering [127]. Potential future directions on logic encryption for HT prevention include expanding logic obfuscation from the chip level to the system level, and the key management in key-based obfuscation schemes [126].

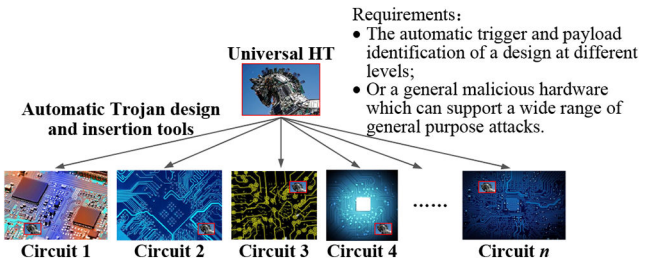

Fig. 4 *Universal HT and automatic Trojan insertion*

Although logic obfuscation is usually used as a DFS method to prevent Trojans, the opposite application is also possible [128, 129]. Vijayakumar *et al.* [128] indicated that physical design obfuscation can also be used to insert parametric Trojans. Such an example is demonstrated by Becker *et al.* [7], where the dopant polarities of transistors are changed to insert HT while making the HT difficult to be detected.

10.9 FPGA Trojan attacks and defenses

Compared with ASIC HTs, the works on FPGA Trojans are relatively less, both on attacks and defenses. The research of FPGA Trojan is not systematic and comprehensive at present. With the widespread use of FPGAs, the FPGA Trojan research is a valuable research direction, e.g. [45, 46, 97, 102].

11 Conclusion

HT is an emerging threat to hardware security and information security. In the last decade, a large number of HT detection techniques have been proposed. However, much less research studies have been conducted in the design and implementation of HTs. In this paper, we provide a review of the development of HT implementations in the last decade and also make an outlook. Unlike all previous surveys or most HT works that focus on Trojan detection from the defender's perspective, for the first time, we study the Trojans from an attacker's perspective, focusing on the attacker's methods, capabilities, evading detection techniques, and challenges. We conclude that the HT implantation or HT-related attacks can be launched at any stages of the IC supply chain, including the testing stage and the distribution stage that was rarely discussed in previous works. There are significant differences in the capabilities of attackers at each stage, which can be roughly divided into three levels: level 1, i.e. in-house design team attackers and 3PIP vendor attackers; level 2, i.e. CAD tools attackers, fabrication stage attackers, and testing stage attackers; level 3, i.e. distribution stage attackers. Similar to the ASIC scenario, FPGA Trojan attacks are also feasible at all stages of the FPGA supply chain. Some potential future directions on HT implementation and defense have emerged, which are tit-for-tat endless games. This paper would hopefully help defenders better understand the Trojan insertion to design reliable defense techniques, and better protect the circuits against HT attacks.

12 Acknowledgments

This work was supported by the National Natural Science Foundation of China (no. 61602241) and the Engineering and

Physical Sciences Research Council (EPSRC) (EP/N508664/-CSIT2).

13 References

[1] Bhunia, S., Hsiao, M.S., Banga, M., *et al.*: 'Hardware Trojan attacks: threat analysis and countermeasures', *Proc. IEEE*, 2014, **102**, (8), pp. 1229–1247

[2] Tehranipoor, M., Koushanfar, F.: 'A survey of hardware Trojan taxonomy and detection', *IEEE Des. Test Comput.*, 2010, **27**, (1), pp. 10–25

[3] Chakraborty, R.S., Narasimhan, S., Bhunia, S.: 'Hardware Trojan: threats and emerging solutions'. Proc. IEEE Int. High Level Design Validation and Test Workshop, San Francisco, USA, November 2009, pp. 166–171

[4] Wu, T.F., Ganesan, K., Hu, Y.A., *et al.*: 'TPAD: hardware Trojan prevention and detection for trusted integrated circuits', *IEEE Trans. Comput-Aided Des. Integr. Circuits Syst.*, 2016, **35**, (4), pp. 521–534

[5] Agrawal, D., Baktir, S., Karakoyunlu, D., *et al.*: 'Trojan detection using IC fingerprinting'. Proc. IEEE Symp. on Security and Privacy, Oakland, USA, May 2007, pp. 296–310

[6] Ghandali, S., Becker, G.T., Holcomb, D., *et al.*: 'A design methodology for stealthy parametric Trojans and its application to bug attacks'. Int. Conf. on Cryptographic Hardware and Embedded Systems, Santa Barbara, USA, August 2016, pp. 625–647

[7] Becker, G.T., Regazzoni, F., Paar, C., *et al.*: 'Stealthy dopant-level hardware Trojans'. Int. Workshop on Cryptographic Hardware and Embedded Systems, Santa Barbara, USA, August 2013, pp. 197–214

[8] Adee, S.: 'The hunt for the kill switch', *IEEE Spectr.*, 2008, **45**, (5), pp. 34–39

[9] Yang, K., Hicks, M., Dong, Q., *et al.*: 'A2: analog malicious hardware'. Proc. IEEE Symp. on Security and Privacy, San Jose, USA, May 2016, pp. 18–37

[10] 'The intel management engine: an attack on computer users' freedom'. Available at https://www.fsf.org/blogs/sysadmin/the-management-engine-an-attack-on-computer-users-freedom, 2018

[11] 'Intel x86s hide another CPU that can take over your machine (you can't audit it)', https://boingboing.net/2016/06/15/intel-x86-processors-ship-with.html, 2016

[12] Rostami, M., Koushanfar, F., Karri, R.: 'A primer on hardware security: models, methods, and metrics', *Proc. IEEE*, 2014, **102**, (8), pp. 1283–1295

[13] Jacob, N., Merli, D., Heyszl, J., *et al.*: 'Hardware Trojans: current challenges and approaches', *IET Comput. Digit. Tech.*, 2014, **8**, (6), pp. 264–273

[14] Karri, R., Rajendran, J., Rosenfeld, K., *et al.*: 'Trustworthy hardware: identifying and classifying hardware Trojans', *IEEE Comput.*, 2010, **43**, (10), pp. 39–46

[15] Shakya, B., He, T., Salmani, H., *et al.*: 'Benchmarking of hardware Trojans and maliciously affected circuits', *J. Hardware Syst. Secur.*, 2017, **1**, (1), pp. 85–102

[16] Baumgarten, A., Steffen, M., Clausman, M., *et al.*: 'A case study in hardware Trojan design and implementation', *Int. J. Inf. Secur.*, 2011, **10**, (1), pp. 1–14

[17] Zhang, J., Xu, Q.: 'On hardware Trojan design and implementation at register-transfer level'. Proc. IEEE Int. Symp. on Hardware-Oriented Security and Trust, Austin, USA, June 2013, pp. 107–112

[18] Fern, N., San, I., Koç, C.K., *et al.*: 'Hiding hardware Trojan communication channels in partially specified SoC bus functionality', *IEEE Trans. Comput-Aided Des. Integr. Circuits Syst.*, 2017, **36**, (9), pp. 1435–1444

[19] Subraman, K.S., Antonopoulos, A., Abotabl, A.A., *et al.*: 'Demonstrating and mitigating the risk of an FEC-based hardware Trojan in wireless networks', *IEEE Trans. Inf. Forensic Secur.*, 2019, **14**, (10), pp. 2720–2734

[20] Tsoutsos, N.G., Konstantinou, C., Maniatakos, M.: 'Advanced techniques for designing stealthy hardware Trojans'. Proc. ACM Annual Design Automation Conf., San Francisco, USA, June 2014, pp. 1–4

[21] Jin, Y., Kupp, N., Makris, Y.: 'Experiences in hardware Trojan design and implementation'. Proc. IEEE Int. Workshop on Hardware-Oriented Security and Trust, San Francisco, USA, July 2009, pp. 50–57

[22] Elnaggar, R., Chakrabarty, K., Tahoori, M.B.: 'Hardware Trojan detection using changepoint-based anomaly detection techniques', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 2019, **27**, (12), pp. 2706–2719

[23] Kumaki, T., Yoshikawa, M., Fujino, T.: 'Cipher-destroying and secret-key-emitting hardware Trojan against AES core'. Proc. IEEE Int. Midwest Symp. on Circuits and Systems, Columbus, USA, August 2013, pp. 408–411

[24] King, S.T., Tucek, J., Cozzie, A., *et al.*: 'Designing and implementing malicious hardware'. Proc. USENIX Workshop on Large-Scale Exploits and Emergent Threats, San Francisco, USA, April 2008, pp. 1–8

[25] Santos, J.C.M., Fei, Y.: 'Designing and implementing a malicious 8051 processor'. Proc. IEEE Int. Symp. on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, Austin, USA, October 2012, pp. 63–66

[26] Reece, T., Limbrick, D.B., Wang, X., *et al.*: 'Stealth assessment of hardware Trojans in a microcontroller'. Proc. IEEE Int. Conf. on Computer Design, Montreal, Canada, September 2012, pp. 139–142

[27] Karri, R., Rajendran, J., Rosenfeld, K.: 'Trojan taxonomy', in Tehranipoor, M., Wang, C. (Eds.): '*Introduction to hardware security and trust*' (Springer, USA, 2012), pp. 325–338

[28] Dash, P., Perkins, C., Gerdes, R.M.: 'Remote activation of hardware Trojans via a covert temperature channel'. Int. Conf. on Security and Privacy in Communication Systems, Dallas, USA, October 2015, pp. 294–310

[29] Gallais, J.F., Großschädl, J., Hanley, N., *et al.*: 'Hardware Trojans for inducing or amplifying side-channel leakage of cryptographic software'. Int. Conf. on Trusted Systems, Beijing, China, December 2011, pp. 253–270

[30] Reece, T., Robinson, W.H.: 'Analysis of data-leak hardware Trojans in AES cryptographic circuits'. Proc. IEEE Int. Conf. on Technologies for Homeland Security, Boston, USA, November 2013, pp. 467–472

[31] Sturton, C., Hicks, M., Wagner, D., *et al.*: 'Defeating UCI: building stealthy and malicious hardware'. Proc. IEEE Symp. on Security and Privacy, Berkeley, USA, May 2011, pp. 64–77

[32] Liu, Y., Jin, Y., Nosratinia, A., *et al.*: 'Silicon demonstration of hardware Trojan design and detection in wireless cryptographic ICs', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 2017, **25**, (4), pp. 1506–1519

[33] Lin, L., Burleson, W., Paar, C.: 'MOLES: malicious off-chip leakage enabled by side-channels'. Proc. Int. Conf. on Computer-Aided Design, San Jose, USA, November 2009, pp. 117–122

[34] Fyrbiak, M., Wallat, S., Swierczynski, P., *et al.*: 'HAL-The missing piece of the puzzle for hardware reverse engineering, Trojan detection and insertion', *IEEE Trans. Dependable Secur. Comput.*, 2018, **16**, (3), pp. 498–510

[35] 'The coq proof assistant'. Available at https://coq.inria.fr/, 2019

[36] Kaji, S., Kinugawa, M., Fujimoto, D., *et al.*: 'Data injection attack against electronic devices with locally weakened immunity using a hardware Trojan', *IEEE Trans. Electromagn. Compat.*, 2018, **61**, (4), pp. 1115–1121

[37] Bhasin, S., Danger, J.L., Guilley, S., *et al.*: 'Hardware Trojan horses in cryptographic IP cores'. Workshop on Fault Diagnosis and Tolerance in Cryptography, Alamitos, USA, August 2013, pp. 15–29

[38] Kumar, R., Jovanovic, P., Burleson, W., *et al.*: 'Parametric Trojans for fault-injection attacks on cryptographic hardware'. Workshop on Fault Diagnosis and Tolerance in Cryptography, Busan, South Korea, September 2014, pp. 18–28

[39] Shiyanovskii, Y., Wolff, F., Rajendran, A., *et al.*: 'Process reliability based Trojans through NBTI and HCI effects'. Proc. Conf. on Adaptive Hardware and Systems, Anaheim, California, June 2010, pp. 215–222

[40] Lin, L., Kasper, M., Güneysu, T., *et al.*: 'Trojan side-channels: lightweight hardware Trojans through side-channel engineering'. Int. Workshop on Cryptographic Hardware and Embedded Systems, Lausanne, Switzerland, September 2009, pp. 382–395

[41] Xue, M., Bian, R., Liu, W., *et al.*: 'Defeating untrustworthy testing parties: a novel hybrid clustering ensemble based golden models-free hardware Trojan detection method', *IEEE Access*, 2019, **7**, pp. 5124–5140

[42] Yasin, M., Sinanoglu, O., Rajendran, J.: 'Testing the trustworthiness of IC testing: an oracle-less attack on IC camouflaging', *IEEE Trans. Inf. Forensic Secur.*, 2017, **12**, (11), pp. 2668–2682

[43] Swierczynski, P., Fyrbiak, M., Koppe, P., *et al.*: 'Interdiction in practice-hardware Trojan against a high-security USB flash drive', *J. Cryptogr. Eng.*, 2017, **7**, (3), pp. 199–211

[44] Chakraborty, R.S., Saha, I., Palchaudhuri, A., *et al.*: 'Hardware Trojan insertion by direct modification of FPGA configuration bitstream', *IEEE Des. Test*, 2013, **30**, (2), pp. 45–54

[45] Swierczynski, P., Fyrbiak, M., Koppe, P., *et al.*: 'FPGA Trojans through detecting and weakening of cryptographic primitives', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2015, **34**, (8), pp. 1236–1249

[46] Krieg, C., Wolf, C., Jantsch, A.: 'Malicious LUT: a stealthy FPGA Trojan injected and triggered by the design flow'. Proc. 35th Int. Conf. on Computer-Aided Design, Austin, USA, November 2016, pp. 1–8

[47] Marchand, C., Francq, J.: 'Low-level implementation and side-channel detection of stealthy hardware Trojans on field programmable gate arrays', *IET Comput. Digit. Tech.*, 2014, **8**, (6), pp. 246–255

[48] Zheng, J.X., Chen, E., Potkonjak, M.: 'A benign hardware Trojan on FPGA-based embedded systems'. 22nd Int. Conf. on Field Programmable Logic and Applications, Oslo, Norway, August 2012, pp. 464–470

[49] 'Trust-hub'. Available at http://www.trust-hub.org/, 2019

[50] Wang, X., Salmani, H., Tehranipoor, M., *et al.*: 'Hardware Trojan detection and isolation using current integration and localized current analysis'. Proc. IEEE Int. Symp. on Defect and Fault Tolerance of VLSI Systems, Boston, USA, October 2008, pp. 87–95

[51] Li, J., Lach, J.: 'At-speed delay characterization for IC authentication and Trojan horse detection'. Proc. IEEE Int. Workshop on Hardware-Oriented Security and Trust, Anaheim, USA, June 2008, pp. 8–14

[52] Hu, K., Nowroz, A.N., Reda, S., *et al.*: 'High-sensitivity hardware Trojan detection using multimodal characterization'. Proc. Conf. on Design, Automation and Test in Europe, Grenoble, France, March 2013, pp. 1271–1276

[53] Xiao, K., Zhang, X., Tehranipoor, M.: 'A clock sweeping technique for detecting hardware Trojans impacting circuits delay', *IEEE Des. Test*, 2013, **30**, (2), pp. 26–34

[54] Narasimhan, S., Du, D., Chakraborty, R.S., *et al.*: 'Hardware Trojan detection by multiple-parameter side-channel analysis', *IEEE Trans. Comput.*, 2013, **62**, (11), pp. 2183–2195

[55] Nowroz, A.N., Hu, K., Koushanfar, F., *et al.*: 'Novel techniques for high-sensitivity hardware Trojan detection using thermal and power maps', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2014, **33**, (12), pp. 1792–1805

[56] Xue, M., Liu, W., Hu, A., *et al.*: 'Detecting hardware Trojan through time domain constrained estimator based unified subspace technique', *IEICE Trans. Inf. Syst.*, 2014, **97-D**, (3), pp. 606–609

[57] Xue, M., Hu, A., Li, G.: 'Detecting hardware Trojan through heuristic partition and activity driven test pattern generation'. Proc. Communications Security Conf., Beijing, China, May 2014, pp. 1–6

[58] Fujimoto, D., Nin, S., Hayashi, Y.I., *et al.*: 'A demonstration of a HT-detection method based on impedance measurements of the wiring around ICs', *IEEE Trans. Circuits Syst. II, Express Briefs*, 2018, **65**, (10), pp. 1320–1324

[59] Nguyen, L.N., Cheng, C.L., Prvulovic, M., *et al.*: 'Creating a backscattering side channel to enable detection of dormant hardware Trojans', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 2019, **27**, (7), pp. 1561–1574

[60] Chakraborty, R.S., Wolff, F., Paul, S., *et al.*: 'MERO: a statistical approach for hardware Trojan detection'. Int. Workshop on Cryptographic Hardware and Embedded Systems, Lausanne, Switzerland, September 2009, pp. 396–410

[61] Huang, Y., Bhunia, S., Mishra, P.: 'Scalable test generation for Trojan detection using side channel analysis', *IEEE Trans. Inf. Forensic Secur.*, 2018, **13**, (11), pp. 2746–2760

[62] Banga, M., Hsiao, M.S.: 'Trusted RTL: Trojan detection methodology in pre-silicon designs'. Proc. IEEE Int. Symp. on Hardware-Oriented Security and Trust, Anaheim, USA, June 2010, pp. 56–59

[63] Waksman, A., Suozzo, M., Sethumadhavan, S.: 'FANCI: identification of stealthy malicious logic using Boolean functional analysis'. Proc. ACM SIGSAC Conf. on Computer and Communications Security, Berlin, Germany, November 2013, pp. 697–708

[64] Zhang, X., Tehranipoor, M.: 'RON: an on-chip ring oscillator network for hardware Trojan detection'. Proc. Conf. on Design, Automation and Test in Europe, Grenoble, France, March 2011, pp. 1–6

[65] Rajendran, J., Jyothi, V., Sinanoglu, O.*, et al.*: 'Design and analysis of ring oscillator based design-for-trust technique'. Proc. IEEE VLSI Test Symp., Dana Point, USA, May 2011, pp. 105–110

[66] Hicks, M., Finnicum, M., King, S.T.*, et al.*: 'Overcoming an untrusted computing base: detecting and removing malicious hardware automatically'. Proc. IEEE Symp. on Security and Privacy, Oakland, USA, May 2010, pp. 159–172

[67] Nahiyan, A., Xiao, K., Yang, K.*, et al.*: 'AVFSM: A framework for identifying and mitigating vulnerabilities in FSMs'. Proc. 53rd Annual Design Automation Conf. (DAC), Austin, USA, June 2016, pp. 1–6

[68] Xiao, K., Forte, D., Tehranipoor, M.: 'A novel built-in self-authentication technique to prevent inserting hardware Trojans', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2014, **33**, (12), pp. 1778–1791

[69] Shi, Q., Tehranipoor, M.M., Forte, D.: 'Obfuscated built-in self-authentication with secure and efficient wire-lifting', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2018, **38**, (11), pp. 1981–1994

[70] Patnaik, S., Ashraf, M., Sinanoglu, O.*, et al.*: 'A modern approach to IP protection and Trojan prevention: split manufacturing for 3D ICs and obfuscation of vertical interconnects', *IEEE Trans. Emerg. Top. Comput.*, 2019, pp. 1–18, Early access

[71] Li, M., Yu, B., Lin, Y.*, et al.*: 'A practical split manufacturing framework for Trojan prevention via simultaneous wire lifting and cell insertion', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2018, **38**, (9), pp. 1585–1598

[72] Zhao, H., Kwiat, L., Kwiat, K.A.*, et al.*: 'Applying chaos theory for runtime hardware Trojan monitoring and detection', *IEEE Trans. Dependable Secur. Comput.*, 2020, **17**, (4), pp. 716–729

[73] Wang, X., Narasimhan, S., Krishna, A.*, et al.*: 'Sequential hardware Trojan: side-channel aware design and placement'. Proc. IEEE Int. Conf. on Computer Design, Amherst, USA, October 2011, pp. 297–300

[74] Rajendran, J.J.V., Sinanoglu, O., Karri, R.: 'Building trustworthy systems using untrusted components: a high-level synthesis approach', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 2016, **24**, (9), pp. 2946–2959

[75] Zhang, X., Tehranipoor, M.: 'Case study: detecting hardware Trojans in third-party digital IP cores'. Proc. IEEE Int. Symp. on Hardware-Oriented Security and Trust, San Diego, USA, June 2011, pp. 67–70

[76] Jou, J.Y., Liu, C.N.J.: 'Coverage analysis techniques for HDL design validation'. Proceedings of 6th Asia Pacific Chip Design Languages: APCHDL'99, Fukuoka, Japan, October 1999, pp. 48–55

[77] Salmani, H., Tehranipoor, M.: 'Analyzing circuit vulnerability to hardware Trojan insertion at the behavioral level'. Proc. IEEE Int. Symp. on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, New York, USA, October 2013, pp. 190–195

[78] Zhang, J., Yuan, F., Xu, Q.: 'Detrust: defeating hardware trust verification with stealthy implicitly-triggered hardware Trojans'. Proc. ACM SIGSAC Conf. on Computer and Communications Security, Scottsdale, USA, November 2014, pp. 153–166

[79] Love, E., Jin, Y., Makris, Y.: 'Proof-carrying hardware intellectual property: a pathway to trusted module acquisition', *IEEE Trans. Inf. Forensic Secur.*, 2012, **7**, (1), pp. 25–40

[80] Jin, Y., Makris, Y.: 'A proof-carrying based framework for trusted microprocessor IP'. Proc. Int. Conf. on Computer-Aided Design, San Jose, USA, November 2013, pp. 824–829

[81] Veeranna, N., Schäfer, B.C.: 'Hardware Trojan detection in behavioral intellectual properties (IP's) using property checking techniques', *IEEE Trans. Emerg. Top. Comput.*, 2017, **5**, (4), pp. 576–585

[82] Rajendran, J., Dhandayuthapany, A.M., Vedula, V.*, et al.*: 'Formal security verification of third party intellectual property cores for information leakage'. Proc. Int. Conf. on VLSI Design, Kolkata, India, January 2016, pp. 547–552

[83] Liu, C., Rajendran, J., Yang, C.*, et al.*: 'Shielding heterogeneous MPSoCs from untrustworthy 3PIPs through security-driven task scheduling', *IEEE Trans. Emerg. Top. Comput.*, 2014, **2**, (4), pp. 461–472

[84] Narasimhan, S., Chakraborty, R.S., Chakraborty, S.: 'Hardware IP protection during evaluation using embedded sequential Trojan', *IEEE Des. Test Comput.*, 2012, **29**, (3), pp. 70–79

[85] Pilato, C., Basu, K., Shayan, M.*, et al.*: 'High-level synthesis of benevolent Trojans'. Proc. Conf. on Design, Automation and Test in Europe Conf. and Exhibition, Florence, Italy, March 2019, pp. 1124–1129

[86] Pilato, C., Basu, K., Regazzoni, F.*, et al.*: 'Black-hat high-level synthesis: myth or reality?', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 2019, **27**, (4), pp. 913–926

[87] Basu, K., Saeed, S.M., Pilato, C.*, et al.*: 'CAD-base: an attack vector into the electronics supply chain', *ACM Trans. Des. Autom. Electron. Syst.*, 2019, **24**, (4), pp. 38:1–38:30

[88] Bao, C., Forte, D., Srivastava, A.: 'On reverse engineering-based hardware Trojan detection', *IEEE Trans. Comput-Aided Des. Integr. Circuits Syst.*, 2016, **35**, (1), pp. 49–57

[89] Xue, M., Wang, J., Hu, A.: 'An enhanced classification-based golden chips-free hardware Trojan detection technique'. Proc. IEEE Asian Hardware-Oriented Security and Trust, Yilan, Taiwan, December 2016, pp. 1–6

[90] Bian, R., Xue, M., Wang, J.: 'Building trusted golden models-free hardware Trojan detection framework against untrustworthy testing parties using a novel clustering ensemble technique'. Proc. IEEE Int. Conf. on Trust, Security and Privacy in Computing and Communications, New York, USA, July 2018, pp. 1458–1463

[91] Contreras, G.K., Rahman, M.T., Tehranipoor, M.: 'Secure split-test for preventing IC piracy by untrusted foundry and assembly'. IEEE Int. Symp. on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, New York, USA, October 2013, pp. 196–203

[92] Rahman, M.T., Forte, D., Shi, Q.*, et al.*: 'CSST: preventing distribution of unlicensed and rejected ICs by untrusted foundry and assembly'. IEEE Int. Symp. on Defect and Fault Tolerance in VLSI Nanotechnology Systems, Amsterdam, The Netherlands, October 2014, pp. 46–51

[93] Zhang, D., Wang, X., Rahman, M.T.*, et al.*: 'An on-chip dynamically obfuscated wrapper for protecting supply chain against IP and IC piracies', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 2018, **26**, (11), pp. 2456–2469

[94] Jyothi, V., Rajendran, J.J.V.: 'Hardware Trojan attacks in FPGA and protection approaches', in Bhunia, S., Tehranipoor, M. (Eds.): '*The hardware Trojan war: Attacks, myths, and defenses*' (Springer, Switzerland, 2018), pp. 345–368

[95] Mal-Sarkar, S., Krishna, A., Ghosh, A.*, et al.*: 'Hardware Trojan attacks in FPGA devices: threat analysis and effective countermeasures'. Proc. Great Lakes Symp. on VLSI, Houston, USA, May 2014, pp. 287–292

[96] Söll, O., Korak, T., Muehlberghuber, M.*, et al.*: 'EM-based detection of hardware Trojans on FPGAs'. Proc. IEEE Int. Symp. on Hardware-Oriented Security and Trust, Arlington, USA, May 2014, pp. 84–87

[97] Chen, Z., Guo, S., Wang, J.*, et al.*: 'Toward FPGA security in IoT: a new detection technique for hardware Trojans', *IEEE Internet Things J.*, 2019, **6**, (4), pp. 7061–7068

[98] Pino, Y., Jyothi, V., French, M.: 'Intra-die process variation aware anomaly detection in FPGAs'. IEEE Int. Test Conf., Seattle, USA, October 2014, pp. 1–6

[99] Jyothi, V., Thoonoli, M., Stern, R.*, et al.*: 'FPGA trust zone: incorporating trust and reliability into FPGA designs'. Proc. IEEE Int. Conf. on Computer Design, Scottsdale, USA, October 2016, pp. 600–605

[100] Swierczynski, P., Fyrbiak, M., Paar, C.*, et al.*: 'Protecting against cryptographic Trojans in FPGAs'. IEEE Annual Int. Symp. on Field-Programmable Custom Computing Machines, Vancouver, Canada, May 2015, pp. 151–154

[101] Bloom, G., Narahari, B., Simha, R.*, et al.*: 'FPGA SoC architecture and runtime to prevent hardware Trojans from leaking secrets'. Proc. IEEE Int. Symp. on Hardware-Oriented Security and Trust, Washington, USA, May 2015, pp. 48–51

[102] Zhang, Z., Njilla, L., Kamhoua, C.A., *et al.*: 'Thwarting security threats from malicious FPGA tools with novel FPGA-oriented moving target defense', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 2018, **27**, (3), pp. 665–678

[103] Krieg, C., Wolf, C., Jantsch, A.*, et al.*: 'Toggle MUX: how X-optimism can lead to malicious hardware'. Proc. 54th Annual Design Automation Conf. (DAC), Austin, USA, June 2017, pp. 1–6

[104] Cruz, J., Huang, Y., Mishra, P.*, et al.*: 'An automated configurable Trojan insertion framework for dynamic trust benchmarks'. Proc. Conf. on Design, Automation and Test in Europe Conf. and Exhibition, Dresden, Germany, March 2018, pp. 1598–1603

[105] Nasr, A.A., Abdulmageed, M.Z.: 'Automatic feature selection of hardware layout: a step toward robust hardware Trojan detection', *J. Electron. Test.*, 2016, **32**, (3), pp. 357–367

[106] Bao, C., Forte, D., Srivastava, A.: 'On application of one-class SVM to reverse engineering-based hardware Trojan detection'. Int. Symp. on Quality Electronic Design, Santa Clara, USA, March 2014, pp. 47–54

[107] Kulkarni, A., Pino, Y., Mohsenin, T.: 'SVM-based real-time hardware Trojan detection for many-core platform'. Int. Symp. on Quality Electronic Design, Santa Clara, USA, March 2016, pp. 362–367

[108] Kulkarni, A., Pino, Y., Mohsenin, T.: 'Adaptive real-time Trojan detection framework through machine learning'. Proc. IEEE Int. Symp. on Hardware-Oriented Security and Trust, McLean, VA, USA, May 2016, pp. 120–123

[109] Bian, R., Xue, M., Wang, J.: 'A novel golden models-free hardware Trojan detection technique using unsupervised clustering analysis'. Proc. Int. Conf. on Cloud Computing and Security, Haikou, China, June 2018, pp. 634–646

[110] Xue, M., Bian, R., Wang, J.*, et al.*: 'A co-training based hardware Trojan detection technique by exploiting unlabeled ICs and inaccurate simulation models'. Proc. IEEE Int. Conf. on Trust, Security and Privacy in Computing and Communications, New York, USA, August 2018, pp. 1452–1457

[111] Xue, M., Bian, R., Wang, J.*, et al.*: 'Building an accurate hardware Trojan detection technique from inaccurate simulation models and unlabelled ICs', *IET Comput. Digit. Tech.*, 2019, **13**, (4), pp. 348–359

[112] Hasegawa, K., Oya, M., Yanagisawa, M.*, et al.*: 'Hardware Trojans classification for gate-level netlists based on machine learning'. Proc. IEEE Int. Symp. on On-Line Testing and Robust System Design, Sant Feliu de Guixols, Spain, July 2016, pp. 203–206

[113] Elnaggar, R., Chakrabarty, K.: 'Machine learning for hardware security: opportunities and risks', *J. Electron. Test.*, 2018, **34**, (2), pp. 183–201

[114] Clements, J., Lao, Y.: 'Hardware Trojan design on neural networks'. IEEE Int. Symp. Circuits Syst., Sapporo, Japan, May 2019, pp. 1–5

[115] Ye, J., Hu, Y., Li, X.: 'Hardware Trojan in FPGA CNN accelerator'. IEEE 27th Asian Test Symp., Hefei, China, October 2018, pp. 68–73

[116] Odetola, T.A., Mohammed, H.R., Hasan, S.R.: 'A stealthy hardware Trojan exploiting the architectural vulnerability of deep learning architectures: input interception attack (IIA)', arXiv:1911.00783, 2019

[117] Li, W., Yu, J., Ning, X., *et al.*: 'Hu-Fu: hardware and software collaborative attack framework against neural networks'. IEEE Computer Society Annual Symp. on Very Large Scale Integration, Hong Kong, China, July 2018, pp. 482–487
[118] Hu, X., Zhao, Y., Deng, L., *et al.*: 'Practical attacks on deep neural networks by memory Trojaning', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., Early Access*, 2020, pp. 1–14
[119] Becker, G.T., Kasper, M., Moradi, A., *et al.*: 'Side-channel based watermarks for integrated circuits'. Proc. IEEE Int. Symp. on Hardware-Oriented Security and Trust, Anaheim, USA, June 2010, pp. 30–35
[120] Bhunia, S., Tehranipoor, M.: '*The hardware Trojan war: attacks, myths, and defenses*' (Springer, Switzerland, 2017)
[121] Ali, S.S., Chakraborty, R.S., Mukhopadhyay, D., *et al.*: 'Multi-level attacks: an emerging security concern for cryptographic hardware'. Proc. Conf. on Design, Automation and Test in Europe, Grenoble, France, March 2011, pp. 1–4
[122] Chakraborty, R.S., Bhunia, S.: 'Security against hardware Trojan attacks using key-based design obfuscation', *J. Electron. Test.*, 2011, **27**, (6), pp. 767–785
[123] Dupuis, S., Ba, P., Natale, G.D., *et al.*: 'A novel hardware logic encryption technique for thwarting illegal overproduction and hardware Trojans'. Proc. IEEE 20th Int. On-Line Testing Symp., Girona, Spain, July 2014, pp. 49–54
[124] Rathor, V.S., Garg, B., Sharma, G.K.: 'A novel low complexity logic encryption technique for design-for-trust', *IEEE Trans. Emerg. Top. Comput.*, 2018, Early Access, pp. 1–12
[125] Frey, J., Yu, Q.: 'Exploiting state obfuscation to detect hardware Trojans in NoC network interfaces'. Proc. IEEE 58th Int. Midwest Symp. on Circuits and Systems, Fort Collins, USA, August 2015, pp. 1–4
[126] Yu, Q., Dofe, J., Zhang, Z.: 'Exploiting hardware obfuscation methods to prevent and detect hardware Trojans'. Proc. IEEE 60th Int. Midwest Symp. on Circuits and Systems, Boston, USA, August 2017, pp. 819–822
[127] Hoque, T., Yang, K., Karam, R., *et al.*: 'Hidden in plaintext: an obfuscation-based countermeasure against FPGA bitstream tampering attacks', *ACM Trans. Des. Autom. Electr. Syst.*, 2020, **25**, (1), pp. 1–32
[128] Vijayakumar, A., Patil, V.C., Holcomb, D.E., *et al.*: 'Physical design obfuscation of hardware: a comprehensive investigation of device and logic-level techniques', *IEEE Trans. Inf. Forensics Secur.*, 2017, **12**, (1), pp. 64–77
[129] Becker, G.T., Fyrbiak, M., Kison, C.: 'Hardware obfuscation: techniques and open challenges', in Bossuet, L., Torres, L. (Eds.): '*Foundations of hardware IP protection*' (Springer, Cham, 2017), pp. 105–123