Trojans in Early Design Steps—An Emerging Threat

Ilia Polian*, Georg T. Becker†, and Francesco Regazzoni‡
*Universität Passau, Germany
†Horst Görtz Institute for IT Security, Ruhr-Universität Bochum, Germany
‡ALARI, University of Lugano, Switzerland

***Abstract*—Hardware Trojans inserted by malicious foundries during integrated circuit manufacturing have received substantial attention in recent years. In this paper, we focus on a different type of hardware Trojan threats: attacks in the early steps of design process. We show that third-party intellectual property cores and CAD tools constitute realistic attack surfaces and that even system specification can be targeted by adversaries. We discuss the devastating damage potential of such attacks, the applicable countermeasures against them and their deficiencies.**

I. Introduction

Historically, IT security concentrated on attack scenarios targeting software and communication networks, but more recently, the system hardware moved into the focus of attackers. Hardware-related threats are relevant even for extremely software-dominated systems, which still contain some amount of hardware on which the software runs; compromising this hardware makes the entire system vulnerable. Even worse, many software-centric security solutions rely on a hardware-based *root of trust* which stores secret keys and provides essential security functions; successful attacks on such root-of-trust blocks renders the entire security concept ineffective. With the emergence of paradigms like cyberphysical systems, internet of things, or Industrie 4.0 that unify the physical world, IT systems and global connectivity, hardware blocks are at risk to become the Achille's heel of entire infrastructures.

This paper considers one emerging attack scenario: Hardware Trojans. These are malicious modification of system hardware with the purpose to gain control over its functionality and, e.g., be able to deactivate the affected block at the attacker's will ("kill switch"), or establish a side-channel to access confidential data processed by the device ("backdoor"). The term "hardware Trojans" was traditionally associated with threats stemming from external, untrusted foundries. However, this paper is specifically concerned with Trojans that are introduced into the system during early design steps by a rogue in-house designer, by an external provider of intellectual property blocks integrated into the design, or even by a computer-aided desing (CAD) tools. An under-investigated attack surface is the system specification which is created in a lengthy and complex process. If an attacker succeeds in planting a Trojan during the specification phase, such a Trojan is extremely hard to detect, because any trusted reference is completely lacking.

The remainder of this paper is organized as follows. In the next section, background information on hardware Trojans is given. Two underestimated classes of threats: Trojans that directly affect the circuit specification and Trojans in CAD tools, are discussed in Sections III and IV, respectively. Known countermeasures and detection methods are reviewed in Section V, and their applicability and efficacy to Trojans in early design steps is evaluated. Section VI concludes the paper.

II. Background

The term "hardware Trojans" historically referred to threats associated with outsourcing of advanced semiconductor manufacturing into potentially untrustworthy countries [1]. The foundry would receive circuit layout information (in GDS II or a similar format) and, possibly, further design data, in order to produce masks and manufacture the circuit. The alleged threat consists in an intentional—and malicious—modification of the circuit by the foundry before manufacturing, such that the fabricated chip has critical deviations from its layout as provided by its author.

However, the modern understanding of hardware Trojans includes all malicious modifications of a circuit during its creation: by an untrusted foundry (as discussed above), but also by rogue in-house designers, by third-party intellectual-property (3PIP) cores, and by CAD tools. Some early publications refer to the threats in early design steps (before manufacturing), by names like "malicious insertions" or "rogue silicon" [2] but recent publications use the name "hardware Trojans" to include both pre-manufacturing and foundry-level scenarios [3].

Fig. 1 shows the standard integrated circuit design flow and indicates sources of possible Trojan threats. Moreover, the lower part of Fig. 1 shows both: standard quality-assurance techniques which have the potential to identify Trojans, and, in blue, approaches specifically designed to counter Trojans. Table I provides a more detailed overview of different types of hardware Trojans. In the following, we discuss the current understanding of both foundry-level Trojans and Trojans in early design steps, whereas subsequent sections will treat under-investigated classes of hardware Trojans in more detail.

A. Foundry-level hardware Trojan threats

Suppose that a sensitive industrial or military system has a restriction of its geographical location, i.e., it can be operated at its legitimate buyer's site but cannot be moved to a different

TABLE I. Attack surfaces for different types of Trojans

Trojans in circuit specification	
Trojans in early design steps	
Third-party IP core	CAD tools
Trojans in 3PIP core's RTL description	Trojans inserted by high-level synthesis tool
Trojans in 3PIP core's gate-level netlist	Trojans inserted by logic synthesis tool
Trojans in 3PIP core's GDS2 layout	Trojans inserted by physical design tool
Foundry-level Trojans	

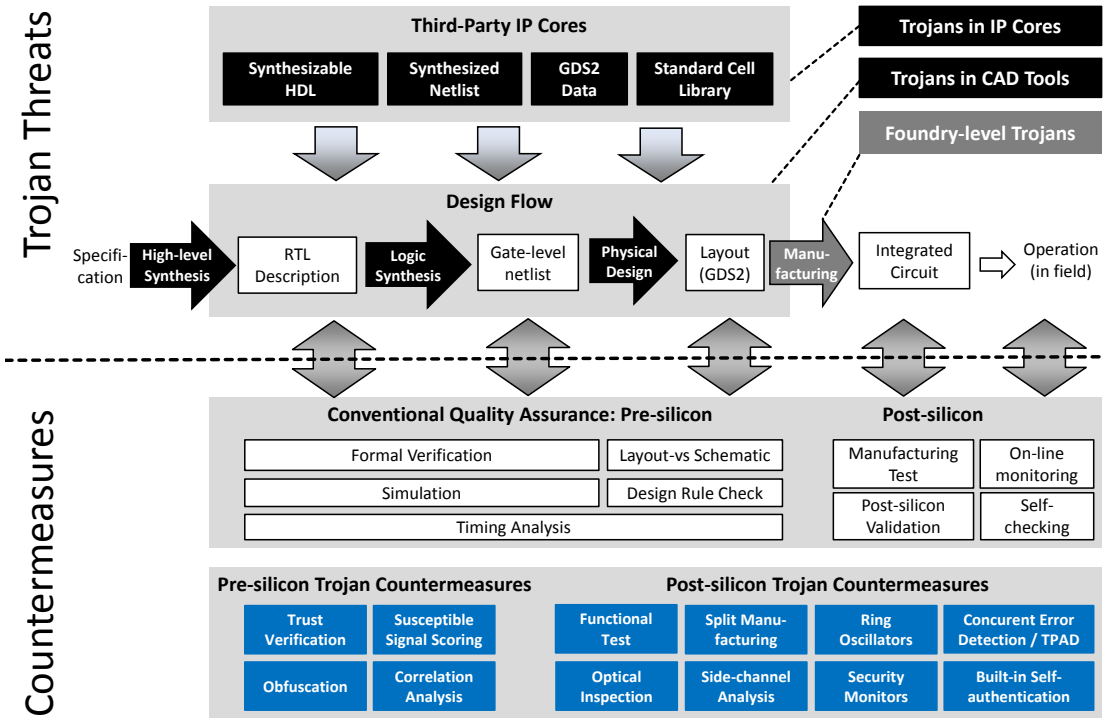

Fig. 1. Overview of circuit design flow, Trojan threats and countermeasures

site or given to a third party. To implement this functionality, one of this system's circuits includes a module that checks GPS coordinates and disables the system when they don't match specification. The foundry-level attacker could find this module by reverse-engineering and deactivate the protection by disconnecting its output—a minimal modification which does not require extensive redesign. A further easy manipulation would be to determine cryptographic blocks that drive an encrypted bus and to disconnect them, such that the bus receives the data in cleartext. Then, an attacker who has physical access to the chip can read out the protected data from the now-unencrypted bus. If the malicious foundry is concerned about legitimate users noticing the absence of encryption, it could replace original crypto modules by self-designed blocks which modify the data in a way that is easily reversed by an attacker.

The key question in this context is *why* a foundry would want to introduce malicious modifications into circuits of their customers. The foundry does not have a direct benefit from deactivation of GPS-powered location checker module. In theory, a potential illegitimate buyer of the equipment in question (e.g., one residing in a country to which export restrictions are in place) could bribe the foundry to deactivate the protection. In practice, it is extremely unlikely that the buyer even knows which foundry will fabricate the circuits for the equipment of interest, and the sheer number of factors that have to fit together for this scenario to happen render it more a conspiracy theory than a realistic threat.

In case of the information-leakage Trojan via deactivation of bus encryption, the foundry itself could later buy systems that incorporate their manipulated circuits, extract the protected data and use them (e.g., resell this data). Obviously, this activity, when carried out on large scale, will attract attention and will likely lead to discovery of the manipulation. The outcome will be enormous legal claims against both the foundry and the individual(s) responsible and a complete collapse of the foundry's reputation. In contrast to other hardware security threats which can be performed by individual, anonymous attackers (like extracting secret data from smart cards by side-channel analysis), foundries are relatively few in number and inseparably attached to their equipment that cannot be quickly relocated. As a consequence, all Trojan manipulations are extremely risky for the foundry, because it would be pretty easy to hold accountable. While it would be wrong to rule out their possibility by this argument, not a single fully documented case was reported by now (the manipulation in [4] turned out to be an undocumented test-access feature rather than an attack).

B. *Early-design-step hardware Trojan threats*

One of the first hardware Trojan demonstrations was a manipulation of the fuel rod position control block in a nuclear reactor [5]. The Trojan affected the hardware module which forwards the temperature readings, thus making the reactor unstable despite redundancy and failsafe mechanisms. King et al. [6] developed techniques to disable memory protection in a microprocessor and introduce a "shadow mode" where full-privilege instructions could run invisible to software. Based on these mecanisms, they demonstrated privilege escalation, login backdoor and password stealing. Hicks et al. [7] demonstrated footholds for unauthorised entering of the supervisor mode, granting unpriviliged malicious software access to protected memory, and allowing the attacker to inject and execute arbitrary code. Wang et al. [8] reported six Trojans triggered by specific instruction/operand sequence which either leaked protected information (protected software code or secret keys) or created malfunctions.

The authors of these papers did not provide a thorough analysis how exactly the Trojans would be inserted. In fact, they assumed that a rogue in-house designer could have done the manipulations. However, these Trojans could also be inserted by a malicious EDA tool (discussed in the next section) or be part of a 3PIP core. A number of Trojan taxonomies [9, 3] distinguish different types of Trojans by their trigger (activation condition) and payload (malicious activity performed upon activation). The trigger condition can consist in a combinational rare-event (low-probability signal assignment) [10], a sequence of input values or operations [11] or in an analog conditions such as lowered voltage [12] or aging [13]. The payload can be roughly categorized into denial-of-service attacks ("kill switch"), which disable the circuit or cripple some of its functions (e. g., reduce the quality of a random-number generator [14]), active manipulations to alter or add its functionality [15]), and information leakage. The latter include Trojans that assist in side-channel analysis of cryptographic hardware [16, 12], or communicate confidential information (including but not restricted to cryptographic secret key) via covert or side-channels like transient power [17] or amplitude of the transmitted wireless signal [18].

III. Trojans in Circuit Specifications

Design specifications are often discussed by several players (customers, hardware and software designers, domain experts, marketing specialists). These players usually come from different communities, each one using a specific "language", often hard to be completely understood by other players. Several efforts have been carried out in the past to analyze and formalize the collection of design specification, using languages such as UML [19], but the analysis of requirements and the finalization of design specifications is still, very often, a somewhat chaotic process. It is thus very likely that a modification inserted in this phase would be pretty hard to identify. Furthermore, all the following design and verification step will directly depend on the specification. As a result, a Trojan inserted during specifications would be almost impossible to detect.

For these reasons, circuit specification seems to be an ideal design step for the insertion of a Trojan. However, to date, this attack scenario is largely unexplored. Trojans inserted during circuit specifications can have mainly three goals. The adversary modifies the specifications to directly include a Trojan circuit, to significantly simplify the insertion of Trojans in following design steps, or to reduce the capabilities of Trojan detection techniques. In the following part of the section we will detail these scenarios.

A. Trojans directly inserted during specification phase

This is the most straightforward attack. The adversary simply modifies the specifications to weaken the overall system, for instance by reducing the number of key updates in a smart card to facilitate a side-channel attacker, by selecting compromised libraries for the design, or by adding debug ports which can be later used to tamper with the device content. More advanced Trojans which can be inserted during specifications are the ones which might look like a device feature. An example can be the insertion of additional circuitry, which looks like the design's watermark but instead allows to leak information about the processed data via a side channel or another covert channel.

B. Simplify Trojan insertion in following steps

Malicious modification of specifications might also have the goal of enabling (or, at least, simplifying) the insertion of a Trojan in subsequent steps of the design process. This can be achieved, for instance, by adding extra circuits which an adversary could modified later to leak information. An example can be the addition of an LFSR to an encryption engine, in order to reveal its functionality. This additional block would justify the addition of extra PADs to the circuit and would allow a malicious designer, in a following step of the design, to connect these PADs also to few bits of the secret key and convey them directly to the adversary.

C. Reduce detection capability

Specifications can even be modified to significantly reduce the detection capabilities applied to the finalized circuit. For instance, an adversary which aims at fooling a Trojan detection algorithm which identifies paths very rarely executed can alter the circuit specification to make appear normal that the Trojan circuit is rarely activated. During the Trojan detection phase, the Trojan circuit will be identified as a rarely executed path. However, since this would match the specification, it would be ignored and the Trojan would remain undetected.

IV. Trojan Toolchain

The majority of scientific results reported so far focused on Trojans inserted by malicious foundries or found in 3PIP cores. While it has been observed that malicious CAD software can create hardware Trojans [20], this scenario was largely ignored in the following years. We argue that CAD tool based hardware Trojan insertion is indeed an absolutely realistic scenario. CAD software has nearly perfect control over the circuit and can add malicious circuitry in an automatic and stealthy manner. In the following, we will first discuss the motivation of a potential adversary to mount a hardware Trojan attack using a CAD tool as a vehicle. Then we will narrow down technical challenges when executing such an attack and formulate requirements for a successful tool-backed hardware Trojan insertion.

A. Why would a CAD tool be malicious?

CAD tools are an integral part of every practical design flow, and even companies which heavily rely on manual circuit optimization make intensive use of automatic tools in every design step. These tools can be roughly divided into synthesis tools working on different levels (shown by black arrows in Fig. 1) and analysis tools for tasks like verification, timing analysis or power estimation. In practice, synthesis and analysis tools are deeply intertwined. The market for such tools is currently divided between three major tool vendors, a large number of small commercial companies typically offered one or very few highly specialized tools, and some amount of free and other non-commercial software. In context of hardware Trojan threats, it is important that most of these tool give their users a very far-reaching flexibility to access their internal data structures via special scripting languages, including SKILL.

When evaluating the possibility of a CAD tool being malicious, the most obvious option is that the company which develops and markets this tool would intentionally incorporate adversarial capabilities into their product. At least for the major providers of tools which are widely used by many customer, the risk of detection is quite high, and its potential consequences would be devastating for the affected tool vendor and for the design-automation industry as a whole. If design houses conclude that they cannot trust CAD tools available on the market, they will either develop their own tools or support development of alternative tools, including open-source tools. These risks can be considered a viable deterrence at least for the larger tool vendors.

Similar argumentation holds for possible political pressure—or legal compulsion—to implement malicious functionality on request of national secret services. While such speculations gained momentum after Snowden revelations, the political and economical consequences of such requirements, should they become public, would be enormous. Companies located in different countries would likely break ties with the affected tool vendors, and governments would support the creation of national tool vendors independent of other countries's services. Such events can spark a spiral of protective measures, sanctions and countersanctions, in the circuit design area and beyond, at the cost of economic efficiency, international collaboration and scientific progress.

Even though the argumentation so far indicates that tool vendors themselves are unlikely to provide malicious versions of their software to customers, attackers other than the tool vendors can still produce such adversarial software without the vendor's consent. This does not mean that an attacker has to develop from scratch a version of the software that is identical to the authentic one except that it adds hardware Trojans to circuit it creates or optimizes. As was mentioned above, most CAD tools are configured, customized and scripted. Changing the default settings may already be sufficient for an attack, as explained in Section IV-B. Moreover, they are modular, and exchanging one module by a malicion variant may easily go undetected.

Once a manipulated version of the CAD tool exists, the attacker must make sure that customers actually use this version and not the Trojan-free software available from the vendor. This can be achieved by social engineering, involving a malicious distributor and/or salesperson. Even more realistic would be a "traditional" computer security breach, where the attacker penetrates the corporate network of the tool-user company and replaces the Trojan-free software by the malicious CAD tool. Once in the network, an attacker could perform further detrimental actions, e.g., replace legitimate 3PIP cores in the customer's database by versions that include hardware Trojans or install a software Trojan which monitors and manipulates the design process (e.g., prevents verification software from being called in order to prevent Trojan detection). Attacks on corporate networks are routinely reported, and in many cases their initiators remain unidentified. Therefore, the "cyberattack" scenario is associated with far less deterrence than malicious tools being provided directly by their vendors. While the attacker will require some degree of knowledge about the internal data organization of the victim company, a competent attacker will be able to obtain this information, possibly from the company's current or past employees. In general, this scenario is realistic and has full damage potential.

B. The technical aspect: How would a Trojan-CAD tool work?

Engineering a malicious CAD tool such that it can reliably insert powerful hardware Trojans into arbitrary circuits without being detected is a much more challenging task that inserting a Trojan into a given circuit, e.g., and 3PIP core. The solution must be fully automatic and completely robust, i.e., no trial-and-error strategies are acceptable. A complete solution for a malicious CAD tool necessitates the following three components:

1) Detection algorithm
2) Payload generator
3) Insertion and hiding procedure

The exact functionality of the three parts depends on the type of Trojan to be employed. For example, if a malicious synthesis tool attempts to plant a Trojan which leaks secret keys out of cryptographic blocks, the tool must first decide whether the circuit being synthesized indeed includes crypto blocks and find where they are located. This activity is related to reverse engineering, except that RE is usually done by a combination of automatic and manual techniques whereas the manipulated tool must locate the Trojan site without any external help.

Once the tool decided whether and when to insert a Trojan, its payload must be generated. For example, if the Trojan will store the secret key in an additional memory which can be later read out, the tool must add this memory to the crypto engine in a manner that will not be detected during subsequent quality-assurance steps. For example, the tool could search for an embedded memory already present in the block and replace it by a slightly larger embedded memory; the extra cells will be used for key storage. If the Trojan is triggered (e.g., by application of a specific plaintext to the circuit's inputs), this trigger must also be generated and added to the circuit.

Since most industrial designs go through a thorough validation, verification and testing process (indicated in the lower part of Fig. 1), Trojan insertion must be accompanied by hiding and obfuscation methods to prevent Trojan detection during quality-assurance procedures. This can be achieved by attacking test procedures simultaneously, e.g., replacing simulation testbenches used for validation by testbenches that never excite the added Trojan functionality, or simply skippiing certain steps (e.g., layout-vs.-schematic checks when Trojans are inserted on layout level during place & route). If vectors used for postmanufacturing testing excite the Trojan (and lead to invalied test responses), the expected responses must be replaced by consistent values.

It is possible to combine CAD tool and 3PIP core Trojans in a simple manner. If malicious versions of 3PIP cores are available on the corporate network, the malicious CAD tool may attempt to replace original, Trojan-free cores by manipulated versions. For example, assume that a core realizes encryption using a balanced design style such as to prevent information leakage exploitable by power analysis. If an attacker can plant a functionally equivalent version of

the core which performs exactly the same encryption but without side-channel countermeasures into the victim network, the malicious tool could simply include the unprotected core instead of the protected version.

Using the three above-mentioned building blocks, a very powerful malicious CAD tool can be constructed that can automatically infect a large number of different designs. But how to construct these building blocks in detail has not been explored so far. Designing such a flexible and powerful tool is quite complex but would be particularly dangerous. Therefore, more research is needed to better understand the danger of malicious CAD tools and develop both technical as well as legal/behavioral countermeasures.

V. Detection and Countermeasures

Circuit design can be understood as a sequence of synthesis steps, from specification down to manufactured silicon, where each step takes a higher-level description and generates a lower-level description. These steps should be consistent, i.e., the output of each synthesis step should either be equivalent to its input, or it should refine it. Insertion of a Trojan breaks up this consistency, and Table II (which follows the structure of Table I) summarizes which abstraction levels are still correct and which are Trojan-affected, depending on their insertion method.

TABLE II. Abstraction levels affected by different Trojans

Type of Trojan Attack		Abstraction levels affected by the Trojan				
		Spec	**RTL**	**Gate**	**Layout**	**Silicon**
Trojans in specification		Trojan	Trojan	Trojan	Trojan	Trojan
Early-design-step						
Trojans in 3PIP core	Trojans in CAD tool					
RTL	High-level synth.	OK	Trojan	Trojan	Trojan	Trojan
Netlist	Logic synthesis	OK	OK	Trojan	Trojan	Trojan
Layout	Physical design	OK	OK	OK	Trojan	Trojan
Foundry-level Trojans		OK	OK	OK	OK	Trojan

A. Countermeasures against foundry-level hardware Trojans

Foundry-level Trojans do not modify any design description, and the complete design stack is consistent. Therefore, pre-silicon methods such as simulation or formal equivalence checking, will not indentify these Trojans and post-silicon methods such as optical inspection [21], functional test [22, 23, 24] or side-channel analysis [25, 26, 27] must be used. Note, however, that it is very much possible to use pre-silicon information to assist in later Trojan detection by, e.g., determining locations where the attacker will likely insert Trojan triggers [28, 10], or to incorporate features that will complicate Trojan insertion [29] or assist in their detection [30]. A very powerful technique to detect foundry-level Trojans is run-time monitoring of either circuit functionality [31] or its parametric behavior [32, 33].

B. Countermeasures against Trojans in early design steps

When considering the middle part of Table II, one unexpected finding is that Trojans in 3PIP cores and in EDA tools are equivalent from detection point of view. For example, Trojans in a gate-level netlist of a 3PIP core and Trojans inserted by a malicious logic-synthesis tool lead to conceptually identical situations, namely that the circuit's RTL description is consistent with its specification but not with its gate-level netlist and lower abstraction levels derived from it. Therefore, any test or verification method that checks consistency (equivalence or refinement) between one the levels above the Trojan insertion point and one of the levels below can potentially detect the Trojan. While there is a plethora of such verification and test methods (indicated in Fig. 1), we will now discuss their potential and shortcomings.

Out of the scenarios in Table II, layout-level Trojans are perhaps the least problematic threat. Efficient and scalable automatic layout-vs-schematic (LvS) tools are available and widely used. They establish a formal equivalence between the gate-level description and the GDS II layout, so any malicious deviations will be reported. Parts of the circuit which are exempt from LvS (like analog blocks) are usually not well suitable for Trojan insertion and can be checked manually. Moreover, design rule checks (DRC) should identify any dopant-level manipulation attempts [14, 12] (note that these techniques were designed for foundry-level, not design-level attacks).

Trojans in gate-level descriptions can be identified by simulation or formal equivalence checking. Simulation is an inherently incomplete technique, and a Trojan with a non-trivial activation condition (e.g., certain instruction sequence in a microprocessor) may go undetected. A successful formal equivalence check would prove the absence of (functional) Trojans, however, state-of-the-art tools are not applicable to arbitrary-sized circuits or require abstraction techniques during which Trojan functionality might be simplified away.

Trojans inserted at the highest abstraction level, namely into an RTL description, appear to be hardest from the detection point of view. The only option to detect them is to check the RTL description against the specification. However, specification is, in practice, often incomplete and lacks important details (such as cycle-accurate execution models). State-of-the-art high-level verification tools cannot handle full-chip descriptions if they include analog or microelectromechanical blocks. Much of practical validation effort is conducted via largely manual techniques such as HDL code review, which provide no guarantee that a Trojan hidden in the code will not be overlooked.

In the ideal case, a specification is accompanied by a comprehensive set of formal properties which can be automatically checked. A Trojan that leads to violation of at least one of these properties will be detected. At the same time, it is extremely hard to prove absence of a system mis-behavior that is not known in advance. Approaches which create a complete property-based description of a circuit which covers its entire functionality [34] might become a viable solution in the future.

Run-time monitoring, which is expensive but effective against foundry-level Trojans [31], does not cover early-design-step Trojans, because checking structures are generated when the circuit's Boolean function is known, i.e., at RTL or gate level. While it is possible to generate check conditions directly out of specification (like "monitoring triangles" in [35]), such conditions are not guaranteed to comprehensively cover the circuit functionality.

C. Countermeasures against Trojans in specification

Counteract Trojan inserted during specification is extremely hard, since generally there is no "golden" specification to compare with (once specifications are in place, it is possible to compare the original one and detect eventual modifications, but this is not possible while specifications are still under preparation). The use of formal approached and languages for requirements collection and system specification, would certainly help to avoid malicious changes of system specification since would allow to use formal methods to verify the properties. However, the most straightforward and effective way to guarantee the correctness of specification is a careful review of the specification documents by the whole team involved in the design.

VI. Conclusion

In this paper we discuss the potential of Trojan inserted in the early steps of design process. We focused on third-party intellectual property cores, CAD tools, and on the system specification step, and we discuss the devastating damage potential of such attacks as well as the possible measures which designer can apply to counteract these attacks.

References

[1] S. Adee. *The Hunt for the Kill Switch*. http://spectrum.ieee.org/semiconductors/design/the-hunt-for-the-kill-switch. 2008.

[2] M. Banga and M. S. Hsiao. "VITAMIN: Voltage Inversion Technique to Ascertain Malicious Insertions in ICs". In: *HOST*. IEEE Computer Society, 2009, pp. 104–107.

[3] S. Bhunia et al. "Hardware Trojan Attacks: Threat Analysis and Countermeasures". In: *Proceedings of the IEEE* 102.8 (2014), pp. 1229–1247.

[4] S. Skorobogatov and C. Woods. "Breakthrough Silicon Scanning Discovers Backdoor in Military Chip". In: *CHES*. Vol. 7428. Lecture Notes in Computer Science. Springer, 2012, pp. 23–40.

[5] S. C. Smith and J. Di. "Detecting Malicious Logic Through Structural Testing". In: *IEEE Region 5 Tech. Conf.* IEEE Computer Society, 2007, pp. 217–222.

[6] S. T. King et al. "Designing and Implementing Malicious Hardware". In: *LEET*. USENIX Association, 2008.

[7] M. Hicks et al. "Overcoming an Untrusted Computing Base: Detecting and Removing Malicious Hardware Automatically". In: *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2010, pp. 159–172.

[8] X. Wang et al. "Software exploitable hardware Trojans in embedded processor". In: *DFT*. IEEE Computer Society, 2012, pp. 55–58.

[9] M. Tehranipoor and F. Koushanfar. "A Survey of Hardware Trojan Taxonomy and Detection". In: *IEEE Design & Test of Computers* 27.1 (2010), pp. 10–25.

[10] S. Dupuis et al. "New testing procedure for finding insertion sites of stealthy hardware Trojans". In: *DATE*. ACM, 2015, pp. 776–781.

[11] S. Narasimhan et al. "TeSR: A robust Temporal Self-Referencing approach for Hardware Trojan detection". In: *HOST*. IEEE, 2011, pp. 71–74.

[12] R. Kumar et al. "Parametric Trojans for Fault-Injection Attacks on Cryptographic Hardware". In: *Fault Diagnosis and Tolerance in Cryptography*. 2014, pp. 18–28.

[13] Y. Shiyanovskii et al. "Process reliability based Trojans through NBTI and HCI effects". In: *AHS*. IEEE, 2010, pp. 215–222.

[14] G. T. Becker et al. "Stealthy Dopant-Level Hardware Trojans". In: *Cryptographic Hardware and Embedded Systems – CHES 2013*. Vol. 8086. 2013, pp. 197–214.

[15] J. Zhang et al. "VeriTrust: Verification for Hardware Trust". In: *IEEE Trans. on CAD of Integrated Circuits and Systems* 34.7 (2015), pp. 1148–1161.

[16] S. Ali et al. "Multi-level attacks: An emerging security concern for cryptographic hardware". In: *DATE*. IEEE, 2011, pp. 1176–1179.

[17] L. Lin, W. Burleson, and C. Paar. "MOLES: Malicious off-chip leakage enabled by side-channels". In: *ICCAD*. ACM, 2009, pp. 117–122.

[18] Y. Liu, Y. Jin, and Y. Makris. "Hardware Trojans in wireless cryptographic ICs: silicon demonstration & detection method evaluation". In: *ICCAD*. IEEE, 2013, pp. 399–404.

[19] International Organization for Standardization. *ISO/IEC 19501: information technology-open distributed processing-unified modeling language (UML) version 1.4. 2*. ISO, 2005.

[20] M. Potkonjak. "Synthesis of trustable ICs using untrusted CAD tools". In: *DAC*. ACM, 2010, pp. 633–634.

[21] S. Bhasin et al. "Hardware Trojan Horses in Cryptographic IP Cores". In: *Fault Diagnosis and Tolerance in Cryptography*. 2013, pp. 15–29.

[22] M. Banga et al. "Guided test generation for isolation and detection of embedded Trojans in ICs". In: *ACM Great Lakes Symposium on VLSI*. ACM, 2008, pp. 363–366.

[23] R. S. Chakraborty et al. "MERO: A Statistical Approach for Hardware Trojan Detection". In: *CHES*. Vol. 5747. Lecture Notes in Computer Science. Springer, 2009, pp. 396–410.

[24] M. Banga and M. S. Hsiao. "Trusted RTL: Trojan Detection Methodology in Pre-silicon Designs". In: *HOST*. IEEE Computer Society, 2010, pp. 56–59.

[25] J. Aarestad et al. "Detecting Trojans Through Leakage Current Analysis Using Multiple Supply Pad I_{DDQ}s". In: *IEEE Trans. Information Forensics and Security* 5.4 (2010), pp. 893–904.

[26] Y. Alkabani and F. Koushanfar. "Consistency-based characterization for IC Trojan detection". In: *ICCAD*. ACM, 2009, pp. 123–127.

[27] B. Cha and S. K. Gupta. "Trojan detection via delay measurements: a new approach to select paths and vectors to maximize effectiveness and minimize cost". In: *DATE*. EDA Consortium San Jose, CA, USA / ACM DL, 2013, pp. 1265–1270.

[28] F. G. Wolff et al. "Towards Trojan-Free Trusted ICs: Problem Analysis and Detection Scheme". In: *DATE*. ACM, 2008, pp. 1362–1365.

[29] R. S. Chakraborty and S. Bhunia. "Security against hardware Trojan through a novel application of design obfuscation". In: *ICCAD*. ACM, 2009, pp. 113–116.

[30] K. Xiao and M. Tehranipoor. "BISA: Built-in self-authentication for preventing hardware Trojan insertion". In: *HOST*. IEEE Computer Society, 2013, pp. 45–50.

[31] T. F. Wu et al. "TPAD: Hardware Trojan Prevention and Detection for Trusted Integrated Circuits". In: *IEEE Trans. CAD* Accepted (2016).

[32] S. Narasimhan et al. "Improving IC Security Against Trojan Attacks Through Integration of Security Monitors". In: *IEEE Design & Test of Computers* 29.5 (2012), pp. 37–46.

[33] J. Rajendran et al. "Design and analysis of ring oscillator based Design-for-Trust technique". In: *VTS*. IEEE Computer Society, 2011, pp. 105–110.

[34] J. Urdahl et al. "Path predicate abstraction by complete interval property checking". In: *FMCAD*. IEEE, 2010, pp. 207–215.

[35] A. Waksman and S. Sethumadhavan. "Tamper Evident Microprocessors". In: *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2010, pp. 173–188.