VLSI Supply Chain Security Risks and Mitigation Techniques: A Survey

Bao Liu[1], Gang Qu[2]

Abstract

Hardware is the foundation of security and trust for any security system. However, recent study has revealed that hardware is subject to a number of security risks. Some of the most severe risks come from the VLSI supply chain. Such risks compromise the foundation of any existing security design. In this paper, we present a systematic survey on these security risks and their corresponding mitigation techniques.

Keywords: Security; VLSI; IP Theft; Hardware Trojan.

1. Introduction

A security system is implemented in many layers. Cryptographic algorithms, including symmetric ciphers, public-key ciphers, and hash functions, form a set of primitives that can be used as building blocks to construct security mechanisms that target specific objectives, such as confidentiality, integrity, and availability [1]. Rigorous theoretical analysis and design of cryptosystems and security protocols are achieved only based on certain assumptions of low level implementation. For example, it is typically assumed that implementations of cryptographic computations are ideal "black boxes" whose internals can neither be observed nor interfered with by any malicious entity. Specifically, all the existing cryptographic primitives have proofs of security based on two assumptions: (1) read-proof hardware; that is, hardware that prevents an adversary from reading anything about the information stored within it; and (2) tamper-proof hardware; that is, hardware that prevents an adversary from changing anything in the information stored within it. However, these assumptions are far from reality. Almost all known security attacks on embedded systems target the implementation rather than taking on the computational complexity of breaking a cryptographic primitive employed in a security mechanism [2]. An interesting analogy can be drawn in this regard between a strong cryptographic algorithm and a highly secure lock on the front door of a house. Burglars attempting to break into a house will rarely try all the combinations necessary to pick such a lock; they may break in through windows, break a door at its hinges, or rob the owner of a key as they are trying to enter the house [3].

Further, there is a growing trend in recent years to migrate software-based security solutions to hardware-based security solutions for much enhanced resistance to software-based security attacks. Such systems range from smartcards to specialized secure co-processing boxes, wherein hardware provides the source of security and trust, e.g., concealing confidential data and providing trustworthy computation for a number of security primitives, e.g., platform identification and authentication, identity, key and certificate management, low-level cryptography, I/O access control, safe data storage, and code integrity checking. Famous examples include Trusted Platform Module (TPM) [4], ARM TrustZone [5], Microsoft Next Generation Secure Computing Base [6], and academic secure processors such as XOM [7], CODESSEAL [8], AEGIS [9, 10], REM [11], SP [12] and SPEF [13, 14].

All these security solutions are based on the assumption that hardware is trustworthy in possessing all the desired security properties. However, hardware is subject to a variety of security risks as recent research has revealed, which compromises the foundation of all the existing security designs. In this paper, we present a systematic review on the security risks in a VLSI supply chain and their respective state-of-the-art mitigation techniques. We do not cover relatively well studied areas of physical access-based attacks such as side-channel analysis [15, 16] and fault injection [17, 18, 19]. We further focus on Application-

[1]B. Liu is with the University of Texas at San Antonio, San Antonio, TX 78249.

[2]G. Qu is with the University of Maryland, College Park, MD 20742.

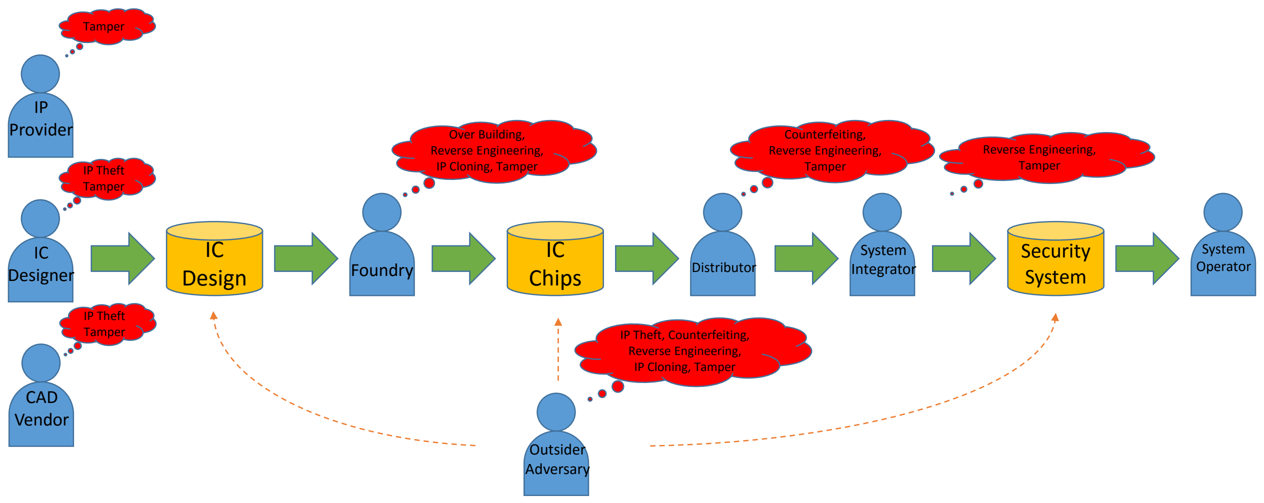
Tamper
IP Provider
IP Theft Tamper
IC Designer
IP Theft Tamper
CAD Vendor
IC Design
Over Building, Reverse Engineering, IP Cloning, Tamper
Foundry
IC Chips
Counterfeiting, Reverse Engineering, Tamper
Distributor
System Integrator
Reverse Engineering, Tamper
Security System
System Operator
IP Theft, Counterfeiting, Reverse Engineering, IP Cloning, Tamper
Outsider Adversary

Figure 1: VLSI supply chain security risks.

Specific Integrated Circuit (ASIC) security, while interested readers may refer to existing literature on FPGA security [20, 21, 22, 23].

The rest of this paper is organized as follows. We present an overview on VLSI supply chain security risks in Section 2 and their state-of-the-art mitigation techniques in Sections 3 and 4. We conclude in Section 6.

2. VLSI Supply Chain Security Risks

Today's semiconductor industry involves multiple business entities on a global scale in design, manufacturing, system integration and distribution of VLSI chips and systems. Without an effective security mechanism, a rogue element in this process - such as an IP provider, an IC design house, a CAD company, a foundry, a distributor or a system integrator - can easily steal design IPs or tamper with an IC design; there is also a possibility that an outsider adversary steals design IPs or tampers with the design (Fig. 1). We categorize such security risks into two groups: (1) *IP theft and misuse*, where an adversary obtains an IP through an illegal channel or uses the authentic IPs illegally, and (2) *IC tamper*, where an adversary modifies the functionality, performance or other features of an IC for various malicious purposes. In terms of the security objectives we mentioned in the introduction, IP theft and misuse compromise IP confidentiality; while IC tamper compromises IC design authenticity and integrity. In the rest of this section, we will elaborate security risks from each of the two categories.

2.1. *IP Theft and Misuse*

IC designs and the intellectual properties (IPs) created during the design process can be protected legally through the means such as patent, copyright, trademark, and trade secret. Design IPs (such as Verilog code, design data, and FPGA configuration bitstream files) can also be encrypted to prevent illegal copy or misuse. However, IP theft is an easy and very profitable business practice due to the lack of effective law enforcement mechanisms, and the need of keeping IP easy to use and reuse. Evidently, we have seen rampant IP theft and misuse in semiconductor industry in recent years. For example, in *over-building*, a contract manufacturer fills an order and continues to build more chips and sell them [23]. In *cloning*, a competitor makes a copy of a design by stealing part or all of a system's intellectual property (IP) [23]. In *reverse engineering*, a competitor extracts not only all the IPs from a design, but also explicit details on how the design works - by package removal, delayering, imaging, circuit extraction and analysis - which allows the IPs to be reused, improved, or disguised to thwart possible legal actions [23, 24].[3]

One common feature of such attacks is that rogue business entities are driven by profit. They are interested in IP theft or misuse rather than IP or IC tamper. For example, an IP licensee may misuse an IP for designs that are not included in the license agreement. This leads to financial loss for the IP owner without necessarily

[3]Despite its potential application in IP theft and IC tampering, reverse engineering is a legal practice, e.g., to collect competitor intelligence, determine patent infringements, and detect hardware Trojans [25].

compromising the authentic design and the end product. Furthermore, profit-driven attacks such as over-building and cloning often happen at a large scale.

2.2. IC Tamper

The end products of IP theft and misuse are often known as *counterfeits*, which are work-alike or cloned products with illegal use of a brand name. Such counterfeits are widespread; the United States Department of Defense has identified more than one million suspect counterfeit parts associated with supply chain compromises in two years [26]. Such counterfeit chips may be made from recycled chips of degraded lifetime, reliability or performance. The most severe form of hardware security risks is that on such counterfeit chips an adversary may tamper with the genuine design and install a "Trojan horse" component which once triggered acts as a logic bomb or information leak back door [27]. An entire Trojan program may be hidden in hardware, e.g., in a Trojan ROM beside a processor (Figure 4) [28]. An adversary may launch such an attack from a foundry, from a system assembly line, or, anyone who captures a hardware device may replace a genuine chip with a counterfeit chip on a printed-circuit board (PCB). A tampered system may still function as expected for minimum footprint, except that it provides a hidden attack mechanism for knowledgeable attackers. Such IC tamper attacks may evade all the existing security solutions implemented at higher (e.g., software application or operating system) levels. For example, the existing static and dynamic code integrity verification techniques detect tamper in the file system, memory or stack rather than a hardware Trojan [7, 9, 10, 12, 29, 30]. As a result, IC tamper attacks compromise a fundamental assumption of the existing security system designs which is the trustworthiness of hardware. They request serious rethinking on security system design.

IC tamper attacks may not lead to obvious profit, while hidden incentives cannot be ruled out, since possible attackers such as amateur hackers, criminal organizations and nation states have different resources, capacities and incentives. In some cases, IC tamper can be an economically viable practice, for example, installing data-collecting hardware spyware. Due to the potential severity of such attacks and the limitations of the existing countermeasure techniques, the Comprehensive National Cyber Security Initiative has identified this supply chain risk management problem as a top national priority [31].

The existing VLSI design and verification techniques are insufficient in mitigating such security risks and ensuring hardware design authenticity, integrity and confidentiality. In today's IC industry, once an IC design is delivered to a foundry, the designers have no effective mechanism to prevent IP theft and misuse. The existing testing techniques only verify if an IC design meets all the specifications. They do not detect the presence of any extra functionality which may be a security backdoor. Table 1 summarizes the VLSI supply chain security risks that we know today and their respective state-of-the-art mitigation techniques. We detail each of these techniques as follows.

Table 1: VLSI supply chain security risks and mitigation techniques.

Security Risks	Mitigation Techniques
Reverse Engineering	Obfuscation
Over-Building	Watermarking, Fingerprinting, Metering
Counterfeiting	Fingerprinting, Metering
Cloning	Watermarking, Fingerprinting
Tamper (IC Design)	Simulation, Formal Verification, Detection, Obfuscation
Tamper (IC Chip)	Reverse Engineering, Testing, Side Channel Analysis, Design for Tamper Detection Design for Tamper Prevention

3. Techniques Against IP Theft and Misuse

3.1. Design Obfuscation

Obfuscation is a long-standing problem in computer security and cryptography as a potentially very powerful tool against reverse engineering. The classic "black-box" definition of obfuscation is to create an implementation of a function f that reveals nothing about f except its input-output behavior. Intuitively, a circuit obfuscator O is an efficient algorithm that, given a circuit C implementing some function f, outputs another circuit $O(C)$ such that (i) (preserving functionality) it computes (perhaps approximately) the same function as f, (ii) (polynomial slowdown) its size is bounded by a given polynomial p in the size of the original circuit, i.e., $|O(C)| \leq p(|C|)$, and, (iii) ("virtual black-box" property) for any efficient adversary that computes some predicate on $O(C)$, there exists an efficient simulator that computes the same predicate with black-box access to an oracle that evaluates f [32, 33, 34].

Recent theoretical studies have shown that, (1) there exist functions that cannot be obfuscated, and (2) there exist functions that can be obfuscated. Barak et al. have shown the existence of (contrived) classes of functions which are not obfuscatable, or, a general purpose obfuscator does not exist [32]. The only positive obfuscation result is of point functions, which are Boolean functions that return logic one on exactly one input, for example, a password checker program. Canetti and Wee separately showed how to obfuscate a point function based on a random oracle, e.g., a hash function that hides all details [33, 35]. An obfuscated point function queries the random oracle on an input, and compares the answer with a stored value. For example, a password checker program encrypts an input, and compares the encryption result with a stored value, which is an encrypted password. As a result, it achieves the virtual black box property of obfuscation. This scheme is based on a weaker definition of obfuscation, which says that there is a negligible probability to distinguish an adversary circuit based on the obfuscated scheme and a simulator based on a black box of the function. As a result, this obfuscation scheme of point functions cannot be extended to obfuscate arbitrary Boolean functions [34], except some specific classes of functions such as d-CNFs [36].

VLSI obfuscation is achievable based on certain special manufacturing technologies, such as split manufacturing, 3D IC integration or embedded reconfigurable logic in ASIC design, which realize the aforementioned "black-box" property. The presence of obfuscated modules does not guarantee that the entire design is obfuscated, as the adversary may still gain knowledge on or tamper with the un-obfuscated part of the design. Design obfuscation methods may not stop an adversary, but certainly increase their cost of reverse engineering. To conclude our discussion on obfuscation, we list representative recently proposed circuit obfuscation techniques.

- In split manufacturing, while the logic gates and the interconnects at the lower metal layers are mass produced at an untrusted foundry, the interconnects at the higher metal layers are customized at a later stage at a trusted site in a semi-customized IC manufacturing technology [37]. An adversary at an untrusted foundry has only black-box access to the upper-layer interconnects. However, an adversary may reconstruct the upper-layer interconnects and the whole design based on subgraph isomorphism [38], and the complexity of doing so can be further reduced based on certain VLSI design objectives, constraints and rules, e.g., minimum wirelength, no combinational loop, and no simultaneous multiple driver of any logic signal [39].

- In 3D IC integration, among several stacked dies mounted on an interposer, a trusted die can be manufactured at a trusted foundry while the other dies and the interposer may be manufactured at an untrusted foundry. An adversary at the untrusted foundry has only black-box access to a trusted die. A trusted die contains logic gates besides interconnects. As a result, the complexity for an adversary to re-construct the whole design is much higher than split manufacturing.

- Similarly, certain modules of an ASIC design can be realized in reconfigurable logic. Such reconfigurable logic can be constructed by a customer or system engineer after the manufacturing and supply process. As a result, any supply chain adversary has only black-box access to it [40, 41].

- In IC camouflaging, multiple logic gates are fabricated in identical or similar layout patterns [42, 43]. While even without high resolution microscopy equipment, an adversary can re-construct a logic netlist including camouflaged logic gates at certain computation complexity [44, 45].

- In logic encryption or logic locking, a combinational logic network is augmented by a group of XOR/XNOR logic gates [46, 47], multiplexers combining different logic cones [48], LUTs forming a reconfigurable logic barrier [46], or permuting the logic inputs/outputs [49], such that only applying a specific vector to the augmented inputs leads to the correct logic. Similarly, an FSM can be augmented by a group of extra finite states which form an obfuscated mode, such that only a correct sequence of inputs transit the FSM out of the obfuscated mode and set the FSM to the correct initial state in the normal operation mode [50, 51, 52, 48, 53]. These techniques prevent an adversary from unlawful operation of a device. However, with knowledge on the function of a protected module, e.g., from the design of the rest of the system, an adversary can recover the key, e.g., based on IC testing techniques [40, 41, 54].

Table 3 compares these technqiues in terms of attack resistance (the computation complexity of the problem

Table 2: Comparison of IC design obfuscation techniques by attack resistance and hardware cost.

Obfuscation Techniques	Attack Resistance	Hardware Cost
Split Manufacturing	Low	Low
3D IC	High	Medium
Reconfigurable Logic	High	High
IC Camouflaging	Low	Medium
Logic Locking	Low	Low

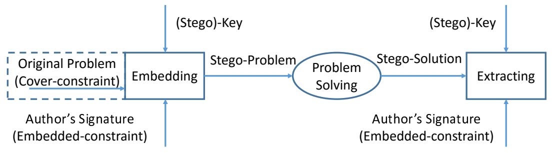
(Stego)-Key
Original Problem (Cover-constraint)
Embedding
Stego-Problem
Problem Solving
Stego-Solution
(Stego)-Key
Extracting
Author's Signature (Embedded-constraint)
Author's Signature (Embedded-constraint)

Figure 2: Constraint-based IC watermark embedding and extraction.

that an attacker must solve to recover the authentic design) and hardware cost.

3.2. *Digital Watermarking*

IP and IC watermarking is to secretly convey the information on content ownership and IP/IC rights. Compared with steganography, IP and IC watermarking further requires the property of robustness, i.e., being infeasible to remove or make useless without destroying the IP/IC at the same time. Watermarking has been applied to protect IPs in all forms, including Verilog codes[55], combinational logic [56, 57], sequential circuits [58], finite state machines [59] and FPGA designs [60], physical design [61], and CAD tools [62]. A survey can be found [63].

Digital watermarking has been widely used for identification, annotation, and copyright of multimedia data such as text, image, audio, and video. Traditional watermarking techniques take advantage of the limitation of human visual and auditory system and embed a signature to an original data set as minute errors. This actually changes the original data and cannot be directly used for the protection of hardware design IPs because the value of design IP relies on its correct functionality and performance. To solve the problem of embedding digital watermark into IC without changing its functionality, a novel constraint-manipulation-based methodology was developed in the late 1990's in UCLA. It was first reported in a series of papers in 1998 [64, 56, 61, 65, 66, 67] and most of the early results can be found in a monograph published in 2003 [68].

A constraint-based watermarking technique translates the to-be-embedded signatures into a set of additional design constraints during the design and implementation of an IP to uniquely encode the signatures into the IP (Figure 2). To hide a signature, the designer first creates another set of constraints using his secret key. These constraints are selected in such a way that they do not conflict with the constraints in the original design. Then the original and additional constraints are combined to form an over-constrained stego-problem. The stego-problem, instead of the original problem, is solved to obtain a stego-solution which has the designer's digital watermark embedded.

During the design and optimization process, the design with these watermarks will have certain specific properties such as constraints. These properties can be extracted from the final design as the proof to the designer's ownership, and the designer can regenerate them using his signature together with his secret key. Cryptography functions such as one-way hash and stream cipher will be used to generate the embedded-constraints. To facilitate the detection of watermarks, a public watermarking scheme was proposed in [69].

Hardware IP watermarking techniques can be categorized as static and dynamic [60, 63, 70]. In static hardware IP watermarking, the watermark is detected without running the IP. The dominant techniques are based on including extra constraints which indicate ownership information in solving an optimization problem [69], such as logic optimization [57], place and route [61]. In dynamic hardware IP watermarking, the watermark can only be detected by running the IP. For example, watermarks can be embedded in logic don't care conditions [55], a watermarked FSM gives the encrypted ownership information for a given input vector sequence [71], or, exhibits a unique property for the input vector sequence which is the encrypted ownership information [58].

3.3. *IC Fingerprinting*

IC watermarking embeds a designer's signature into an IC to claim his ownership and IP/IC rights against IP/IC piracy. Such watermarks do not help forensics such as tracing a copyright violator who distributed illegal copies. The digital fingerprinting techniques solve this problem by embedding a buyer's signature along with a designer's watermark in an IC design. Both the watermark and the fingerprint are invisible identifiers that are embedded in the design permanently for the purpose of IP protection. All copies of an IP share an identical watermark, while each copy of an IP has a unique fingerprint.

Any fingerprinting technique has to address two fundamental problems: (i) how to generate IPs with unique fingerprints effectively and (ii) how to distribute these fingerprinted IPs to the users. While the problem of distributing fingerprinted copies is similar to the well-studied problem of distributing other artifacts such as multimedia data, there are several unique challenges in the IP generation problem: How to create a large amount of copies with no duplicated fingerprints? How to keep the overhead or IP quality degradation at minimum? How to minimize the time and complexity of generating multiple fingerprinted IPs (ideally keeping it close to that of designing a single copy)?

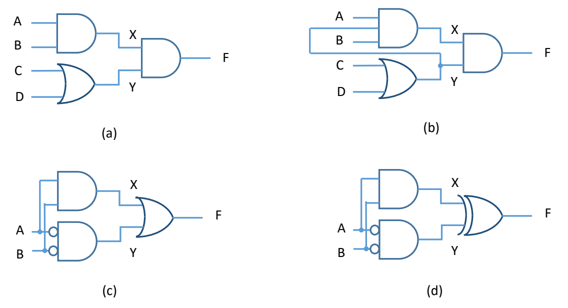

Figure 3: Digital fingerprinting by creating logic equivalent IPs. (a) and (b) differ in an interconnect. (c) and (d) differ in a logic gate with a don't care input vector ($x = y = 1$).

Because of these difficulties, there are much less IC fingerprinting techniques than IC watermarking techniques in literature. Caldwell et al. pioneered in creating fingerprints based on specific VLSI CAD optimization heuristics [70]. They demonstrated IC fingerprinting taking as examples some of the VLSI design related NP-hard problems such as partitioning, Boolean satisfiability (SAT), graph coloring and standard-cell placement problems. These problems are solved by iterative optimization techniques, such that specific constraints similar to those in watermarking can be introduced in iterations to create many unique solutions.

A conceptually-different fingerprinting approach for the graph coloring problem is proposed in [72], where the authors effectively add new constraints to a graph by either adding new edges to create cliques or introducing new nodes and edges to duplicate existing nodes. This increases the solution space such that solving the problem once leads to creation of multiple unique fingerprints. A major issue with this approach and that in [70] is that these techniques must be implemented in an early stage of VLSI design. This leads to significant increase in design time and cost. For instance, each fingerprinted IC must have a different mask, which is impractical given the cost of masks.

Recently reported are two practical fingerprinting methods based on the observability don't cares (ODC) and satisfiability don't cares (SDC) in logic design [73, 74]. For example, insertion of an extra interconnect in Fig. 3 (a) leads to Fig. 3 (b) without changing the Boolean logic function at Y due to the ODC condition [73]; while replacing the OR gate in Fig. 3 (c) by an XOR gate leads to Fig. 3 (d) under a SDC condition [74]. Such changes can be made after IC fabrication, for example, based on reconfigurable logic.

3.4. IC Metering

IC metering targets another critical security problem in IC supply chain: with the asymmetric relationship between an IC design house and an foundry, once a design house delivers a design to a foundry, the design house will have no control on what the foundry can do to the design. This creates the risk of IC over-building wherein an extra volume of chips are fabricated without the design house's permission. "IC metering is a set of security protocols that enable a design house to achieve post-fabrication control over their ICs" [75].

The basic concepts behind IC metering is to embed a unique tag to each IC and make sure that the tag is under the control of the design house instead of the foundry. Many different types of tags have been proposed and used for hardware metering. They can be categorized based on various criteria.

In passive metering, a tag can only be used for chip identification. In active metering, a tag can further enable, disable, or control a chip. Active metering methods can be further classified as internally controlled and external controlled based on whether the control is part of the design. Intrinsic metering does not need any help from additional components or design modification. Extrinsic metering methods do. Depending whether the tag interacts with the chip's functionality, we have non-functional metering and functional metering. Finally, some tags can be reproduced and some cannot which are known as unclonable tags.

The serial number technique is one of the most popular and earliest device tagging technique. A serial number can be physically indented on the device or stored permanently in the memory. These tags are passive, extrinsic, non-functional, and reproducible. The fact that such tags can be reproduced makes it unsuitable to prevent IC over-building.

The ICID tag technique was first proposed in 2007 [76]. In this technique, a sequence of control signals select an array of transistors to drive a capacitive load. The output voltage differs for each chip due to inherent IC manufacturing process variations. Because such variations include random and uncontrollable components, An ICID is considered an unclonable tag and thus can be applied against IC over-building.

The ICID tag technique is the first scheme for generating a weak PUF or random chip ID based on process variations [77]. Other PUF schemes include arbiter, ring oscillator, and SRAM-based [78, 77]. Essentially, a PUF is an (at least partly) disordered physical system P that can be challenged with so-called external stimuli or challenges c, upon which it reacts with

Table 3: Comparison of IC watermarking, fingerprinting and metering techniques by attack resistance, design, verification and hardware costs.

IP-Protecting Techniques	Attack Resistance	Design Cost	Verification Cost	Hardware Cost
IC Watermarking	High	Low	Low/High	Low
IC Fingerprinting	High	High	Low/High	Medium
IC Metering	Low	Low	Medium	Low

corresponding responses r. Contrary to standard digital systems, these responses depend on the micro- or nanoscale structural disorder of the PUF. It is assumed that this disorder cannot be cloned or reproduced exactly, not even by the PUF's original manufacturer, and that it is unique to each PUF. Any PUF P thus implements a unique and individual function f_P that maps challenges c to responses $r = f_P(c)$ [77, 79]. Such a response can be exploited for deriving a standard digital key that is not stored in the hardware and hard to extract, for system identification, or for more complex cryptographic protocols such as oblivious transfer (OT), bit commitment (BC), or key exchange (KE) [77, 79]. A PUF needs to achieve uniqueness, randomness and reliability [80, 78].

3.5. Comparison

Table 3 compares IC watermarking, fingerprinting and metering technqiues. IC watermarks and fingerprints are integrated in IC design, such that an attacker needs to reverse engineer and understand an IC design to remove or forge a watermark or fingerprint. As a result, they have high attack resistance. IC watermarking techniques based on extra design constraints or circuitry have a little design and hardware cost. The cost of IC fingerprinting techniques are higher than IC watermarking techniques because a lot more IC fingerprints are needed for each customer. Dynamic IC watermarking/fingerprinting techniques are easy to verify, while static IC watermarking/fingerprinting techniques require reverse engieering and have a higher cost. IC metering techniques depend on a ICID tag or PUF that is separate from the IC design. As a result, the design and hardware costs for a RFID tag or PUF only count for a small percentage of that of a whole chip. However, their attack resistances need to be examined case by case. For example, R*ü*hrmair et al. discussed many assumptions and limitations of PUF-based techniques in the context of different security protocols [77, 79].

4. Techniques Against IC Tamper

We have two groups of IC tamper detection techniques. The first group of techniques detect tamper for a given IC design. The second group of techniques detect tamper for a given IC chip. We will discuss IC tamper prevention at last.

4.1. IC Design Tamper Detection

This group of techniques address the following problem.

Problem 1 (IC Design Tamper Detection). *Given an IC design (e.g., RTL design in form of Boolean logic expression) and its implementation (e.g., logic design in form of gate-level netlist or layout design), verify that the implementation faithfully realizes the design* without any additional functionality.

A number of existing techniques address this problem, including Layout Versus Schematic (LVS), formal verification and simulation. However, they do not guarantee hardware Trojan detection, and ongoing research is producing new techniques. We elaborate as follow.

4.1.1. Simulation

Simulation is one of the mainstream IC design verification techniques. However, there are a number of limitations in applying simulation techniques for hardware tamper detection. The existing simulation techniques verify an IC design against its specifications; they do not target IC tamper detection or extra functionality identification. A hardware Trojan may perform an extra task without tampering the authentic functionalities. Further, a hardware Trojan may be triggered by a rare event such as power glitch or IC aging which may not even be modeled in a digital system simulation environment. Without a priori knowledge, the likelihood is minimal for a simulator to trigger and detect a hardware Trojan.

4.1.2. Formal Verification

Formal verification verifies if an implementation conforms to its specification by a formal (e.g., mathematical) method [81]. This includes equivalence checking and property checking, e.g., of security requirements such as absence of unprotected path from confidential data. Equivalence checking determines if an implementation realizes no more and no less than what is specified. The "no more" part is exactly needed for Trojan detection. The existing LVS techniques check the equivalence between an IC layout and its schematic design, e.g., based on graph isomorphism. For logic equivalence between a Boolean logic expression and a gate-level netlist, one can represent both in a canonical form, e.g., Ordered Binary Decision Diagram (OBDD), and check the graph isomorphism of the two OBDDs [82]. The OBDD technique achieves a polynomial average runtime for the NP-complete problem which worst case runtime remains exponential. However, the complexity of checking functional equivalence of sequential systems remains very high: two functional equivalent sequential systems may look very different due to retiming optimization and/or different finite state encodings; while the exponential number of state transition paths leads to the state explosion problem. To mitigate this problem, for a finite state machine, one may represent (1) each finite state in a vector of Boolean variables, (2) all the finite states in a Boolean function which returns true for all the finite state representations in Boolean variable vectors, and (3) all the state transitions xRy in a Boolean function with two sets of Boolean variables, one for state x and the other for state y. The OBDD technique can be subsequently applied for equivalence and property checkings [83]. Such techniques are in the category of symbolic model checking which consist of systematically exhaustive exploration of a mathematical model based on smart and domain-specific abstraction techniques. Symbolic model checking techniques are more scalable than explicit-state model checking techniques which enumerate each reachable state. However, their scalability is still limited.

Another category of formal verification techniques are deductive verification. This usually involves describing the subject system and the properties to verify in one of the interactive or automatic theorem provers such as HOL [84], Coq [85], PVS [86], etc. Notable examples include the four color theorem proof which was based on Coq [87]. Recent techniques include the Proof-Carrying Code (PCC) technique wherein software developer/vendors provide proofs for customer-specified safety policies in a binary executable [88], and the similar Proof-Carrying Hardware (PCH) framework which is a SAT solver-based combinational equivalence checker between a design specification and a design implementation on a reconfigurable platform [89, 90], and a new PCH framework which uses the Coq functional language [85] for proof construction and leverages the Coq platform for automatic proof validation [91, 92]. These techniques require that the verification engineer have detailed understanding on the system and the properties to verify and convey them in formal specification.

4.1.3. Redundant Logic and Hard-to-Excite Signal Identification

Even if an implementation is logic equivalent to its original design, a hardware Trojan may still be hidden in redundant logic, and could be activated by fault injection, e.g., based on perturbation of power supply, clock, or injection of an optical fault [19] or an IC aging sensor [28, 93]. To address this problem, techniques such as Unused Circuit Identification (UCI) have been proposed [94] and improved [95]. Further, ATPG techniques can be leveraged to identify redundant or untestable logic [96]. Because hardware Trojans are supposed to be triggered by a rare event, another group of techniques locate hard-to-excite signals as candidates of hardware Trojan trigger [97]. These techniques can be combined. For example, Banga and Hsiao proposed a four-step procedure to locate suspicious logic in third-party IPs: (1) A sequential ATPG technique removes easy-to-detect signals. (2) A full-scan N-detect ATPG technique identifies hard-to-excite and/or propagate signals. (3) To narrow down the list of suspected signals and identify the gates associated with a hardware Trojan, a SAT solver checks equivalence of the suspicious netlist containing the rarely triggered signals against the netlist of the circuit exhibiting correct behavior. (4) Finally, clusters of untestable gates in the circuit were determined using the region isolation approach on the suspected signals list [96]. Zhang and Tehranipoor proposed another multi-stage approach which includes assertion based verification, code coverage analysis, redundant circuit removal, equivalence analysis and use of sequential Automatic Test Pattern Generation (ATPG) for suspicious signals identification [98]. These techniques do not need an authentic design as reference. However, these techniques are limited as a hardware Trojan may not be based on redundant logic or a hard-to-excite signal.

4.2. IC Chip Tamper Detection

This group of techniques address the following problem.

Problem 2 (IC Chip Tamper Detection). *Given an IC design and an IC chip, verify that the IC chip faithfully realizes the design* without any additional functionality.

4.2.1. Reverse Engineering

Part of the technical difficulty of the IC chip tamper detection problem is that a verification engineer may not even know the design details of a chip. Reverse engineering can be applied to extract the design details of a chip, such that the IC design tamper detection techniques in subsection 4.1 can be applied. This method can detect any tamper by a designer, an IP provider, a CAD vendor, a system integrator or a distributor. However, an adversary at an untrusted foundry may tamper with only a few IC chips, while the existing reverse engineering techniques are destructive: traditional IC reverse engineering techniques require decapsulation and passive layer removal, while new techniques such as X-ray microscopy damage transistors [24]. As a result, combination of reverse engineering and IC design tamper detection techniques cannot be applied to all the chips and cannot guarantee detection of hardware tamper by an adversary at a foundry.

4.2.2. *Testing*

To detect a hardware Trojan by testing, (1) the testing procedure must activate the hardware Trojan, and (2) the activated hardware Trojan leads to a behavior deviation of the VLSI system such as an incorrect output that can be observed. However, neither is easy to achieve. Activating a hardware Trojan is very difficult since a hardware Trojan can be triggered by a rare event which is unknown to a test engineer [99, 100, 101]. If the hardware Trojan trigger logic includes an IC aging sensor, the hardware Trojan cannot be activated before the IC is sufficiently aged [28, 93]. Even if a hardware Trojan is activated, the hardware Trojan may still keep a minimum footprint, for example, sending out confidential information in a side channel [102] or by steganography [103] without tampering with the result of any authentic computation in the host system.

4.2.3. *Side Channel Analysis*

Besides IC testing techniques, side channel analysis techniques have been proposed for IC tamper detection. These techniques collect IC characterizations in a side channel such as timing performance [104], power consumption [105], temperature, or electromagnetic emission [106], and find outliers for candidates of tampered chips. These techniques rely on a golden tamper-free reference design which may be achieved by reverse engineering a few chips [106] or by self referencing [107]. However, a few significant problems exist: (1) the significant effect of parametric variations could easily bury the effect of a small hardware Trojan; and (2) it is very difficult to activate a hardware Trojan. Without being activated, a dormant hardware Trojan has very little footprint, e.g., possibly in leakage [108]. These make side channel analysis very difficult.

4.2.4. *IC Design for Tamper Detection*

Due to the limitations in IC testing and side channel analysis, IC design techniques are needed to facilitate tamper detection. For example, ring oscillator-based on-chip sensors are proposed to detect hardware Trojan-induced power supply voltage droop [109, 110]. The built-in self-authentication (BISA) technique leverages the existing built-in self-test (BIST) techniques [109, 111].

Another group of techniques are based on concurrent checking. A variety of concurrent checking techniques are available in the traditional fault-tolerant computing literature [112, 113]. These techniques are appealing for tamper detection because they do not require hardware Trojan activation which is difficult to achieve without a priori knowledge on the hardware Trojan. On the other hand, such techniques are tamper detection but not tamper prevention techniques.

In concurrent checking, a hardware system generates information bits and check bits, e.g., parity bits, duplicate of the information bits as in a dual-module redundancy (DMR) scheme, or in a more efficient error-detecting code (EDC) [112, 113]. Checking the consistency between the information bits and the check bits detects runtime errors such as soft errors or adversary tampers (e.g., triggered by a timer) which cannot be detected by testing.

At system level, fault tolerant processor design includes a variety of redundant execution and concurrent checking techniques [113]. (1) Lockstepping schemes compare internal states (e.g., program control flow [114, 115, 116], hardware control signals [117], memory access [118], and reasonableness of results [119, 120]) in each cycle with duplicated program runs in a watchdog co-processor. (2) Redundant Multi-Threading (RMT) schemes compare only outputs of committed instructions [121, 122, 123, 124, 125]. (3) EDCC-based hardware assertion techniques lead to more hardware-efficient fault tolerant processors compared with lockstepping or RMT [126, 127, 128].

Concurrent checking techniques have been adopted for tamper detection. For example, the TrustNet and DataWatch architectures include on-chip monitors which check the consistency of control signals and data in a microprocessor [129]. Against remote attacks and physical attacks, e.g., wherein an adversary has physical access to the hardware and can tamper with memory busses or instructions and data stored in memory chips, the DEFENSE architecture includes a FPGA which performs runtime concurrent integrity checking besides encryption and decryption for instruction and data blocks

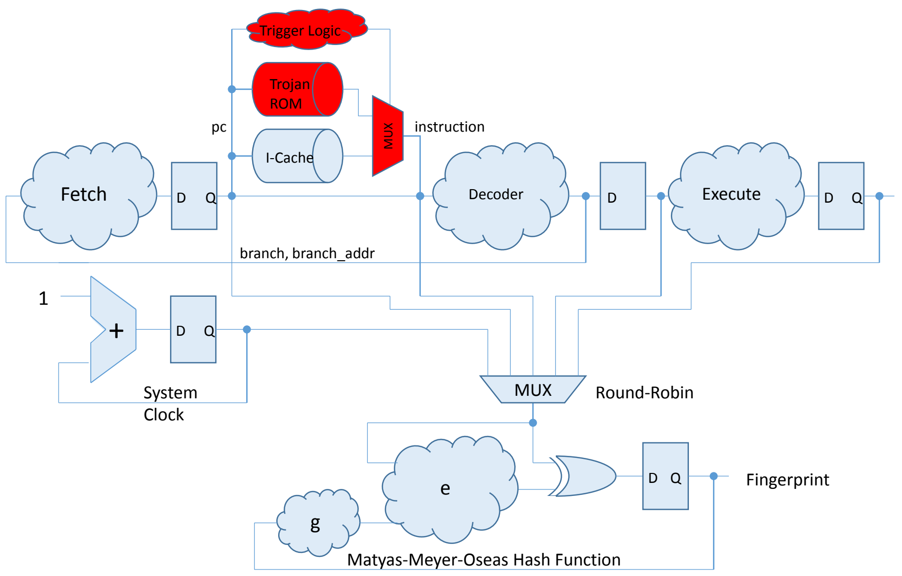

Figure 4: A code injection hardware Trojan including a Trojan ROM, multiplexers, and trigger logic (red), and a tamper-evident architecture including multiplexers that sample runtime signals including the system time in a round-robin scheme, and a fingerprint generator based on the Matyas-Meyer-Oseas hash function (below the instruction pipeline) in a processor.

between a processor core and a memory chip [130]. Fetch-time or runtime integrity checking is included in many secure processors such as AEGIS [9, 10], REM [11], SP [12], and SPEF [13, 14] against remote attacks wherein an adversary performs code injection, reuse or data injection or substitution attacks via a communication channel. Further against IC tamper attacks such as code injection from a hardware Trojan (Fig. 4 (a)), concurrent checking needs to be applied against runtime signals inside an IC chip, and such a checking mechanism needs to be protected from tamper by a supply chain adversary, e.g., based on reconfigurable logic, split manufacturing, or reconfigurable resistive RAM (RRAM) switches [131]. For example, the Tamper-Evident Architecture (TEA) computes a fingerprint or keyed cryptographic hash for runtime signals during the computation in a hardware system, and verify such a fingerprint off-chip for computation integrity verification and malicious program detection (Fig. 4). As a result, a supply chain adversary or hardware Trojan (1) cannot generate correct check bits or fingerprint for a malicious program, and (2) cannot tamper with the checking mechanism [28]. This technique verifies integrity and authenticity of a program run without guaranteeing the integrity and authenticity of the system, e.g., it does not detect a dormant hardware Trojan. The cost of such techniques can be controlled similarly to the existing Design for Testability (DFT) techniques [132].

4.3. IC Tamper Prevention

Besides tamper detection, we further need tamper response (recovery or self-destruction) and tamper evidence (recording and digital forensics) techniques [20]. A harder problem is tamper prevention. A number of techniques have been proposed for tamper prevention with limited effectiveness.

A straightforward solution is IC design obfuscation. However as we know obfuscation of an entire VLSI system is not possible while some modules may be obfuscated such as based on reconfigurable logic or a trusted die [40, 41].

Another technique is to obfuscate data or runtime signals in a hardware system for data confidentiality and tamper prevention. Instruction and data encryption for storage is a common technique, while their decryption brings performance cost [7, 8]. Bus scrambling such as by permutation or XORing with a pseudo-random number achieves only weak cryptographic strength [101], while achieving stronger cryptographic strength comes with significant cost. Private circuit techniques address the problem of achieving data confidentiality in the presence of an attacker who can observe at most t signals in a hardware system at any given time [133, 134]. Fascinating progress has been achieved in the field of homomorphic cryptography [135] and secure multi-party computation [136, 137], allowing arbitrary computation based on encrypted data - albeit at a prohibitive cost for efficient VLSI application.

5. Hardware Security Research Trends

Hardware security is under heated research. Ongoing research development is leading to rapid innovations. We notice several trends in this field: (1) VLSI technology development has made some traditional techniques such as side channel analysis increasingly difficult (higher integration leads to a decreasing signal-to-noise ratio in cutting-edge technologies for side channel analysis). While this also provides opportunities to develop new security solutions based on emerging VLSI technologies, for example, for Truly Random Number Generation (TRNG) and PUFs. (2) New research trends in system integration such as Internet of Things and Cyber-Physical Systems demand security research for such emerging systems. IoT/CPS are complex systems including software, firmware and hardware components. They are expected to be deployed in diverse, dynamic, and potentially hostile environment, such as, for example, an adversary may easily gain physical possession of an IoT/CPS device, and launch hardware attacks. The traditional security research addresses security protocols, primitives, and their software implementations, while hardware is assumed to be trustworthy. This research gap gives rise to recent hardware security research efforts. (3) From a system perspective, new hardware or system security research works need to be based on more realistic attack models which include as many as possible attack methods. (4) Examining hardware security solutions in the context of higher level security primitives and protocols such as in [77, 79] provides new perspectives. (5) Examining security solutions from an economic and/or social perspective is much needed to facilitate security solution deployment in the real world because the cost and benefit of security techniques are ultimately shared by parties in a supply chain, an industry ecosystem, and a society. Security-oriented business management and policy making and enhancement mechanisms are much needed.

6. Summary

Hardware security risks such as from a VLSI supply chain come under scrutiny only recently. Such security risks compromise the foundation of all existing security designs. Consequently, research on their mitigation techniques has been intensive in recent years. In this paper, we present a systematic survey on the hardware security risks from a VLSI supply chain and their state-of-the-art countermeasure techniques. Although significant progress has been made over the years, many important problems remain open and critical solutions missing in this field. We hope that this survey help increase public awareness to the problem and foster further technology development in the field.

Acknowledgements

The authors thank NSF support by grant CISE-1538418, and anonymous reviewers for their comments.

Reference

[1] B. Schneier, Applied Cryptography: Protocols, Algorithms and Source Code in C, John Wiley and Sons, 1996.

[2] S. Ravi, A. Raghunathan, S. Chakradhar, Tamper resistance mechanisms for secure embedded systems, in: Intl. Conf. on VLSI Design, 2004.

[3] B. Schneier, Security Pitfalls in Cryptography, `http://www.schneier.com/essay-pitfalls.html`.

[4] Trusted Computing Group, Trusted Platform Module (TPM) Specifications.
URL `http://www.trustedcomputinggroup.org/resources/tpm_main_specification`

[5] ARM, Building a secure system using trustzone technology, ARM Limited.

[6] Microsoft, Next-generation secure computing base.
URL `http://www.microsoft.com/resources/ngscb/default.mspx`

[7] D. Lie, C. Thekkath, M. Mitchell, et al., Architecture support for copy and tamper resistant software, in: Proc. International Conference on Architecture Support for Programming Languages and Operating Systems (ASPLOS-IX), 2000, pp. 168–177.

[8] O. Gelbart, P. Ott, B. Narahari, R. Simha, A. Choudhary, J. Zambreno, Codesseal: Compiler/fpga approach to secure applications, in: Proc. IEEE Int. Conf. on Intelligence and Security Informatics, 2005, pp. 530–535.

[9] G. E. Suh, D. Clarke, B. Gassend, M. van Dijk, S. Devadas, AEGIS: architecture for tamper-evident and tamper-resistant processing, in: Proc. International Conference on Supercomputing, 2003.

[10] G. E. Suh, C. W. O'Donnell, I. Sachdev, S. Devadas, Design and implementation of a single-chip secure processor using physical random functions, in: Proc. International Symposium on Computer Architecture, 2004.

[11] A. M. Fiskiran, R. B. Lee, Runtime execution monitoring (REM) to detect and prevent malicious code execution, in: Proc. IEEE Intl. Conf. Computer Design, 2004.

[12] R. B. Lee, P. C. S. Kwan, J. P. McGregor, J. Dwoskin, Z. Wang, Architecture for protecting critical secrets in microprocessors, in: Proc. International Symposium on Computer Architecture (ISCA), 2005, pp. 2–13.

[13] D. Kirovski, M. Drinic, M. Potkonjak, Enabling trusted software integrity, in: Proc. 10th Int. Conf. Architecture Support for Programming Languages and Operating Systems, 2002, pp. 108–120.

[14] M. Drinic, D. Kirovski, A hardware-software platform for intrusion prevention, in: Porc. 37th Ann. IEEE/ACM Intl. Symp. Microarchitecture, 2004, pp. 233–242.

[15] P. C. Kocher, Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems, Advances in Cryptology - CRYPTO'96, Lecture Notes in Computer Science V. 1109 (1996) 104–113.

[16] P. Kocher, J. Jaffe, B. Jun, Differential power analysis, in: Proc. Intl. Cryptography Conf. Advances in Cryptography, 1999, pp. 388–397.

[17] H. Choukri, M. Tunstall, Handbook of Information Security, Volume 3, John Wiley and Sons, 2006.

[18] S. Skorobogatov, R. Anderson, Optical fault induction attacks, in: Proc. Cryptographic Hardware and Embedded Systems, 2003, pp. 2–12.

[19] J. G. J. van Woudenberg, M. F. Witteman, F. Menarini, Practical optical fault injection on secure microcontrollers, in: Proceedings of the 2011 Workshop on Fault Diagnosis and Tolerance in Cryptography, 2011, pp. 91–99.

[20] Altera, Anti-tamper capabilities in FPGA designs.
URL `http://www.altera.com/literature/wp/wp-01066-anti-tamper-capabilities-fpga.pdf/`

[21] S. Drimer, Volatile FPGA design security - a survey (2008).
URL `http://www.saardrimer.com/sd410/papers/fpga_security.pdf`

[22] Microsemi, Overview of data security using microsemi FPGAs and SoC FPGAs.
URL `http://www.microsemi.com/document-portal/doc_view/132873-overview-of-data-security-using-microsemi-fpgas-and-soc-fpgas`

[23] Microsemi, Overview of design security using microsemi FPGAs and SoC FPGAs.
URL `http://www.microsemi.com/document-portal/doc_view/132862-overview-of-design-security-using-microsemi-fpgas-and-soc-fpgas`

[24] R. Torrance, D. James, The state-of-the-art in IC reverse engineering, in: Cryptographic Hardware and Embedded Systems - CHES 2009, Vol. 5747 of Lecture Notes in Computer Science, 2009, pp. 363–381.

[25] Chipworks. [link].
URL `http://www.chipworks.com/`

[26] U. S. Senate, Inquiry into counterfeit electronic parts in the department of defense supply chain, report of the committee on armed services (May 21, 2012).

[27] M. Beaumont, B. Hopkins, T. Newby, Hardware trojans prevention, detection, countermeasures (a literature review), DSTO-TN-1012, unclassified, Tech. rep., Australian Government, Department of Defense, Defense Science and Technology Organization, `http://www.dtic.mil/get-tr-doc/pdf?AD=ADA547668` (2011).

[28] B. Liu, R. Sandhu, Fingerprint-based detection and diagnosis of malicious programs in hardware, IEEE Trans. on Reliability.

[29] R. Elbaz, D. Champagne, C. Gebotys, R. B. Lee, N. Potlapally, L. Torres, Hardware mechanisms for memory authentication: A survey of existing techniques and engines, Trans. on Comput. Sci. (2009) 1–22.

[30] R. B. Lee, D. K. Karig, J. P. Mcgregor, Z. Shi, Enlisting hardware architecture to thwart malicious code injection, in: In Proceedings of the 2003 International Conference on Security in Pervasive Computing, 2003, pp. 237–252.

[31] National Security Council, The Comprehensive National Cybersecurity Initiative.
URL `http://www.whitehouse.gov/cybersecurity/comprehensive-national-cybersecurity-initiative`

[32] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, K. Yang, On the (im)possibility of obfuscating programs, in: Proc. International Conference on Cryptography, 2001, pp. 1–18.

[33] R. Canetti, Towards realizing random oracles: Hash functions that hide all partial information, in: Proc. International Conference on Cryptography, 1997, pp. 455–469.

[34] A. Narayanan, V. Shmatikov, On the limits of point function obfuscation (2006).
URL `http://eprint.iacr.org/2006/182`

[35] H. Wee, On obfuscating point functions, in: Proc. ACM Symp. the Theory of Computing, 2005.

[36] Z. Brakerski, G. N. Rothblum, Black-box obfuscation for d-cnfs, Cryptography ePrint Archieve (2013).
URL `http://eprint.iacr.org/2013/557`

[37] IARPA, Trusted integrated chips (TIC), `http://www.iarpa.gov/index.php/research-programs/tic/baa` (2011).

[38] F. Imeson, A. Emtenan, S. Garg, M. V. Tripunitara, Securing computer hardware using 3d integrated circuit (ic) technology and split manufacturing for obfuscation, in: Proc. 22nd USENIX Security Symposium, 2013, pp. 495–510.

[39] J. Rajendran, O. Sinanoglu, R. Karri, Is split manufacturing secure, in: Proc. Conference on Design Automation and Test in Europe, 2013, pp. 1259 – 1264.

[40] B. Liu, B. Wang, Embedded reconfigurable logic for ASIC design obfuscation against supply chain attacks, in: Proc. Conference on Design Automation and Test in Europe, 2014.

[41] B. Liu, B. Wang, Reconfiguration-based vlsi design for security, IEEE Journal on Emerging and Selected Topics in Circuits and Systems.

[42] L. W. Chow, J. P. Baukus, B. J. Wang, R. P. Cocchi, Camouflaging a standard cell based integrated circuit, uS Patent 8,151,235 (2012).
URL `http://www.google.com/patents/US8151235`

[43] SypherMedia, Circuit camouflage technology: SMI IP protection and anti-tamper technologies (2012).
URL `http://www.smi.tv/SMI_SypherMedia_Library_Intro.pdf`

[44] M. E. Massad, S. Garg, M. V. Tripunitara, Integrated circuit (IC) decamouflaging: Reverse engineering camouflaged ICs within minutes, in: Proc. of NDSS, 2015.

[45] J. Rajendran, M. Sam, O. Sinanoglu, R. Karri, Security analysis of integrated circuit camouflaging, in: ACM Conference on Computer and Communications Security, 2013, pp. 709–720.

[46] A. Baumgarten, A. Tyagi, J. Zambreno, Preventing IC piracy using reconfigurable logic barriers, IEEE Design and Test of Computers (2010) 66 – 75.

[47] J. Roy, F. Koushanfar, I. Markov, EPIC: Ending piracy of integrated circuits, in: Proc. Conference on Design Automation and Test in Europe, 2008, pp. 1069 – 1074.

[48] R. S. Chakraborty, S. Bhunia, HARPOON: an obfuscation-based SoC design methodology for hardware protection, IEEE Trans. Computer-Aided Design 28(10) (2009) 1493–1502.

[49] J. A. Roy, F. Koushanfar, I. L. Markov, Protecting bus-based hardware IP by secret sharing, in: Proc. ACM/IEEE Design Automation Conf., 2008, pp. 846 –851.

[50] Y. M. Alkabani, F. Koushanfar, Active hardware metering for intellectual property protection and security, in: Proc. USENIX Security Symposium, 2007, pp. 291 –306.

[51] R. S. Chakraborty, S. Bhunia, Hardware protection and authentication through netlist level obfuscation, in: Proc. IEEE Intl. Conf. Computer-Aided Design, 2008, pp. 674–677.

[52] R. S. Chakraborty, S. Bhunia, Security against hardware Trojan through a novel application of design obfuscation, in: Proc. IEEE Intl. Conf. Computer-Aided Design, 2009, pp. 113–116.

[53] A. R. Desai, M. S. Hsiao, et al., Interlocking obfuscation for anti-tamper hardware, in: CSIIRW, 2012, pp. 1 – 4.

[54] J. Rajendran, Y. Pino, O. Sinanoglu, R. Karri, Security analysis of logic obfuscation, in: Proc. ACM/IEEE Design Automation Conf., 2012, pp. 83 – 89.

[55] L. Yuan, R. Pari, G. Qu, Soft IP protection: Watermarking

HDL source codes, in: 6th Information Hiding Workshop, 2004, pp. 224–238.

[56] A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, G. Wolfe, Watermarking techniques for intellectual property protection, in: Proc. ACM/IEEE Design Automation Conf., 1998.

[57] A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, G. Wolfe, Constraint-based watermarking techniques for design intellectual property protection, IEEE Trans. Computer-Aided Design 20(10) (2001) 1236–1252.

[58] A. L. Oliverira, Robust techniques for watermarking sequential circuit designs, in: Proc. ACM/IEEE Design Automation Conf., 1999, pp. 837–842.

[59] L. Yuan, G. Qu, Information hiding in finite state machine, in: 6th Information Hiding Workshop, 2004, pp. 340–354.

[60] A. K. Jain, L. Yuan, P. R. Pari, G. Qu, Zero overhead watermarking technique for fpga designs, in: Proc. Great Lakes Symp. VLSI, 2003, pp. 147–152.

[61] A. B. Kahng, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, G. Wolfe, Robust ip watermarking methodologies for physical design, in: Proc. ACM/IEEE Design Automation Conf., 1998, pp. 782–787.

[62] L. Yuan, G. Qu, A. Srivastava, VLSI CAD tool protection by birthmarking design solutions, in: Proc. Great Lakes Symp. VLSI, 2005, pp. 341–344.

[63] A. T. Abdel-Hamid, S. Tahar, M. Aboulhamid, A survey on IP watermarking techniques, Design Automation for Embedded Systems 9 (2004) 211–227.

[64] J. Lach, W. H. Mangione-Smith, M. Potkonjak, Fingerprinting digital circuits on programmable hardware, in: Information Hiding Workshop, 1998, pp. 16–31.

[65] J. Lach, W. H. Mangione-Smith, M. Potkonjak, Fpga fingerprinting techniques for protecting intellectual property, in: Custom Integrated Circuits Conference, 1998, pp. 299–302.

[66] J. Lach, W. H. Mangione-Smith, M. Potkonjak, Signature hiding techniques for fpga intellectual property protection, in: Proc. IEEE Intl. Conf. Computer-Aided Design, 1998, pp. 186–189.

[67] G. Qu, M. Potkonjak, Analysis of watermarking techniques for graph coloring problem, in: Proc. IEEE Intl. Conf. Computer-Aided Design, 1998, pp. 190–193.

[68] G. Qu, M. Potkonjak, Intellectual property protection in VLSI designs: theory and practice, Springer Science & Business Media, 2003.

[69] G. Qu, Publicly detectable watermarking for intellectual property authentication in vlsi design, IEEE Trans. Computer-Aided Design 21(11) (2002) 1363–1368.

[70] A. E. Caldwell, H.-J. Choi, A. B. Kahng, S. Mantik, M. Potkonjak, G. Qu, J. L. Wong, Effective iterative techniques for fingerprinting design ip, IEEE Trans. Computer-Aided Design 23(2) (2004) 208–215.

[71] I. Torunoglu, E. Charbon, Watermarking-based copyright protection of sequential functions, IEEE J. Solid State Circuits 35(3) (2000) 434–440.

[72] G. Qu, M. Potkonjak, Fingerprinting intellectual property using constraint-addition, in: Design Automation Conference, 2000.

[73] C. Dunbar, G. Qu, A practical circuit fingerprinting method utilizing observability dont care conditions, in: Proc. ACM/IEEE Design Automation Conf., 2015, pp. 113–118.

[74] C. Dunbar, G. Qu, Satisfiability don't care condition based circuit fingerprinting techniques, in: Proc. Asian and South Pacific Design Automation Conference, 2015.

[75] F. Koushanfar, Hardware metering: A survey, in: M. Tehranipoor, C. Wang (Eds.), Introduction to Hardware Security and Trust, Springer New York, 2012, pp. 103–122.

[76] K. Lofstrom, W. R. Daasch, D. Taylor, IC identification circuit using device mismatch, in: Proc. IEEE Solid State Circuits Conference, 2000, pp. 372–373.

[77] U. R*ü*hrmair, S. Devadas, F. Koushanfar, Security based on physical unclonability and disorder, in: M. Tehranipoor, C. Wang (Eds.), Introduction to Hardware Security and Trust, Springer New York, 2012, pp. 65–102.

[78] C. B*ö*hm, M. Hofer, Physical Unclonable Functions in Theory and Practice, 1st Edition, Springer International Publishing, 2014.

[79] U. R*ü*hrmair, M. van Dijk, PUFs in security protocols: Attack models and security evaluations, in: IEEE Symposium on Security and Privacy, 2013, pp. 286–300.

[80] M. Bhargava, Reliable, secure, efficient physical unclonable functions, Ph.D. thesis, Carnegie Mellon University (2013).

[81] McFarland, Formal verification of sequential hardware, IEEE Trans. Computer-Aided Design 12(5) (1993) 633–654.

[82] R. E. Bryant, Graph-based algorithms for boolean function manipulation, IEEE Trans. Computers C-35(8) (1986) 677 – 691.

[83] K. L. McMillan, Symbolic model checking: An approach to the state explosion problem, Ph.D. thesis, Carnegie Mellon University (1992).

[84] H. Community, HOL interactive theorem prover, `https://hol-theorem-prover.org/`.

[85] INRIA, The coq proof assistant, `http://coq.inria.fr/` (2010).

[86] SRI, PVS specification and verification system, `http://pvs.csi.sri.com/` (2014).

[87] G. Gonthier, Formal proof - the four-color theorem, Notices of the American Mathematical Society 55(11) (2008) 1382 – 1392.

[88] G. C. Necula, Proof-carrying code, in: POPL '97: Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, 1997, pp. 106–119.

[89] S. Drzevitzky, U. Kastens, M. Platzner, Proof-carrying hardware: Towards runtime verification of reconfigurable modules, in: International Conference on Reconfigurable Computing and FPGAs, 2009, pp. 189–194.

[90] S. Drzevitzky, Proof-carrying hardware: Runtime formal verification for secure dynamic reconfiguration, in: Field Programmable Logic and Applications (FPL), 2010 International Conference on, 2010, pp. 255–258.

[91] E. Love, Y. Jin, Y. Makris, Enhancing security via provably trustworthy hardware intellectual property, in: Hardware-Oriented Security and Trust (HOST), 2011 IEEE International Symposium on, 2011, pp. 12–17.

[92] E. Love, Y. Jin, Y. Makris, Proof-carrying hardware intellectual property: A pathway to trusted module acquisition, IEEE Transactions on Information Forensics and Security 7 (1) (2012) 25–40.

[93] Y. Shiyanovskii, F. Wolff, C. Papachristou, D. Weyer, W. Clay, Exploiting semiconductor properties for hardware trojans, ePrint arXiv:0906.3834 (2009).
URL `http://arxiv.org/pdf/0906.3834`

[94] M. Hicks, M. Finnicum, S. T. King, M. M. K. Martin, J. M. Smith, Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically, in: Proc. of IEEE Symposium on Security and Privacy, 2010, pp. 159–172.

[95] C. Sturton, M. Hicks, D. Wagner, S. T. King, Defeating uci: Building stealthy and malicious hardware, in: Proc. of IEEE Symposium on Security and Privacy, 2011, pp. 64–77.

[96] M. Banga, M. Hsiao, Trusted RTL: Trojan detection methodology in pre-silicon designs, in: IEEE International Symposium

on Hardware-Oriented Security and Trust (HOST), 2010, pp. 56–59.

[97] R. S. Chakraborty, F. G. Wolf, S. Paul, C. A. Papachristou, S. Bhunia, MERO: A statistical approach for hardware trojan detection, in: Proc. Cryptographic Hardware and Embedded Systems, 2009, pp. 396–410.

[98] X. Zhang, M. Tehranipoor, Case study: Detecting hardware trojans in third-party digital ip cores, in: Hardware-Oriented Security and Trust (HOST), 2011 IEEE International Symposium on, 2011, pp. 67–70.

[99] R. S. Chakraborty, S. Narasimhan, S. Bhunia, Hardware trojans: Threats and emerging solutions, in: IEEE Intl. High Level Design Validation and Test Workshop, 2009, pp. 166 – 171.

[100] Y. Jin, N. Kupp, Y. Makris, Experience in hardware Trojan design and implementation, in: Proc. IEEE International Workshop on Hardware-Oriented Security and Trust, 2009, pp. 50–57.

[101] A. Waksman, S. Sethumadhavan, Silencing hardware backdoors, in: Proc. IEEE Symp. on Security and Privacy, 2011, pp. 49–63.

[102] L. Lin, M. Kasper, T. G*ü*neysu, C. Paar, W. Burleson, Trojan side-channels: Lightweight hardware Trojans through side-channel engineering, in: Proc. Cryptographic Hardware and Embedded Systems, 2009, pp. 382–395.

[103] F. A. P. Petitcolas, R. J. Anderson, M. G. Kuhn, Information hiding: A survey 87(7) 1062–1078.

[104] Y. Jin, Y. Makris, Hardware Trojan detection using path delay fingerprint, in: Proc. IEEE International Workshop on Hardware-Oriented Security and Trust, 2008, pp. 51–57.

[105] R. Rad, J. Plusquellic, M. Tehranipoor, Sensitivity analysis to hardware trojans using power supply transient signals, 2008, pp. 3–7.

[106] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, B. Sunar, Trojan detection using IC fingerprinting, in: Proc. IEEE Symposium on Security and Privacy, 2007, pp. 296–310.

[107] D. Du, S. Narasimhan, R. S. Chakraborty, S. Bhunia, Self-referencing: a scalable side-channel approach for hardware trojan detection, in: Proc. Cryptographic Hardware and Embedded Systems, 2010.

[108] M. Potkonjak, A. Nahapetian, M. Nelson, T. Massey, Hardware trojan horse detection using gate-level characterization, in: Proc. ACM/IEEE Design Automation Conf., 2009, pp. 688–693.

[109] M. Tehranipoor, H. Salmani, X. Zhang, Integrated Circuit Authentication: Hardware Trojans and Counterfeit Detection, 1st Edition, Springer International Publishing, 2014.

[110] X. Zhang, On-chip structures and techniques to improve the security, trustworthiness and reliability of integrated circuits, Ph.D. thesis, University of Connecticut (2013).

[111] K. Xiao, M. Tehranipoor, Bisa: Built-in self-authentication for preventing hardware trojan insertion, 2013.

[112] M. Göessel, V. Ocheretny, E. Sogomonyan, D. Marienfeld, New Methods of Concurrent Checking, Springer, 2008.

[113] S. Mukherjee, Architecture Design for Soft Errors, Morgan Kaufmann Publishers, 2008.

[114] D. Lu, Watchdog processors and structural integrity checking, IEEE Trans. Computers 31(7) (1982) 681–685.

[115] M. Namjoo, Techniques for concurrent testing of vlsi processor operation, in: Proc. IEEE Intl. Test Conf., 1982, pp. 461–468.

[116] J. P. Shen, M. A. Schuette, On-line self-monitoring using signatured instruction streams, in: Proc. IEEE Intl. Test Conf., 1983, pp. 275–282.

[117] S. F. Daniels, A concurrent test technique for standard microprocessors, in: Dig. Papers Compcon Spring 83, 1983, pp. 389–394.

[118] M. Namjoo, E. J. McCluskey, Watchdog processors and capability checking, in: Dig. Papers 12th Annu. Int. Symp. Fault Tolerant Comput., FTCS-12, 1982, pp. 245–248.

[119] A. Mahmood, E. J. McCluskey, Concurrent error detection using watchdog processors - a survey, IEEE Trans. Computers 37(2) (1988) 160–174.

[120] P. P. Shirvani, E. J. McCluskey, Fault-tolerant systems in a space environment: The CRC ARGOS project, in: CRC Technical Report No. 98-2 (CSL TR No. 98-774), 1998.

[121] T. M. Austin, Diva: A reliable substrate for deep submicron microarchitecture design, in: Proc. Annu. Intl. Symp. on Microarchitecture (MICRO), 1999, pp. 196–207.

[122] D. Bernick, B. Bruckert, P. D. Vigna, D. Garcia, R. Jardine, J. Klecka, J. Smullen, NonStop advanced architecture, in: Proc. Intl. Conf. on Dependable Systems and Networks(DSN), 2005, pp. 12–21.

[123] M. A. Gomaa, C. Scarbrough, T. N. Vijaykumar, I. Pomeranz, Transient fault-recovery for chip multiprocessors, in: Proc. International Symposium on Computer Architecture, 2003, pp. 98–109.

[124] E. Rotenberg, Ar-smt: A microarchitectural approach to fault tolerance in microprocessors, in: Annu. Fault-Tolerant Computing Systems (FTCS), 1999, p. 84.

[125] T. N. Vijaykumar, I. Pomeranz, K. Cheng, Transient fault recovery using simultaneous multithreading, in: Proc. International Symposium on Computer Architecture, 2002.

[126] T. J. Slegel, R. M. Averill, M. A. Check, B. C. Giamei, B. W. Krumm, C. A. Krygowski, W. H. Li, J. S. Liptay, J. D. MacDougall, T. J. McPherson, J. A. Navaroo, E. M. Schwarz, K. Shum, C. F. Web, IBM's S/390 G5 microprocessor design, IEEE Micro (1999) 12–23.

[127] L. Spainhower, T. A. Gregg, IBM S/390 parallel enterprise server G5 fault tolerance: A historical perspective, IBM Journal of Research and Development 43(5/6) (1999) 863–873.

[128] C. Webb, z6 - the next-generation mainframe microprocessor (2007).
URL `http://speleotrove.com/decimal/IBM-z6-mainframe-microprocessor-Webb.pdf`

[129] A. Waksman, S. Sethumadhavan, Tamper evident microprocessors, in: Proc. IEEE Symp. on Security and Privacy, 2010, pp. 1–16.

[130] M. Abramovici, P. Bradley, Integrated circuit security - new threats and solutions, in: Proc. of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies, 2009.

[131] S. Mitra, H.-S. P. Wong, S. Wong, The Trojan-proof chip, IEEE Spectrum 2 (2015) 46 – 51.

[132] M. L. Bushnell, V. D. Agrawal, Essentials of Electronic Testing For Digital, Memory, And Mixed-Signal VLSI Circuits, Norwell, MA: Kluwer, 2000.

[133] Y. Ishai, A. Sahai, D. Wagner, Private circuits: Securing hardware against probing attacks, in: Proc. International Conference on Cryptography, 2003, pp. 463–481.

[134] Y. Ishai, M. Prabhakaran, A. Sahai, D. Wagner, Private circuits ii: Keeping secrets in tamperable circuits, in: Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vol. 4004 of Lecture Notes in Computer Science, Springer, 2006, pp. 308–327.
URL `http://www.iacr.org/cryptodb/archive/2006/EUROCRYPT/2508/2508.pdf`

[135] C. Gentry, A fully homomorphic encryption scheme, Ph.D. thesis, Stanford University (2009).
URL `http://crypto.stanford.edu/craig/`

[136] C. Orlandi, Is multiparty computatoin any good in practice?, in: ICASSP, 2011.

[137] J. Saia, M. Zamani, Recent results in scalable multi-party computation, in: G. Italiano, T. Margaria-Steffen, J. Pokorny, J.-J. Quisquater, R. Wattenhofer (Eds.), SOFSEM 2015: Theory and Practice of Computer Science, Vol. 8939 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2015, pp. 24–44.