ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Μέθοδοι επαλήθευσης και ταυτοποίησης ολοκληρωμένων κυκλωμάτων για λόγους ασφαλείας

Διπλωματική Εργασία

Πιτσιάβας Κωνσταντίνος

Επιβλέπων: Σταμούλης Γεώργιος

Βόλος 2023

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Formal Security Verification and Identification methods of Integrated Circuits

Diploma Thesis

Pitsiavas Konstantinos

Supervisor: Stamoulis Georgios

Volos 2023

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

«Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής».

Ο Δηλών

Κωνσταντίνος Πιτσιάβας
8/7/2023

Περίληψη

Η ποιότητα του υλικού του υπολογιστή θεωρείται συχνά δεδομένη από τους προγραμματιστές λογισμικού και συστημάτων. Οι εξελίξεις στην τεχνολογία επεξεργασίας ημιαγωγών και η παγκοσμιοποίηση της ροής σχεδίασης System-on-chip (SoC) έχουν δώσει στους απατεώνες παρόχους πνευματικής ιδιοκτησίας (IP) την ευκαιρία να εισάγουν κακόβουλα κυκλώματα (γνωστά και ως Trojans υλικού) στις IP τους.

Αυτή η διατριβή παρουσιάζει απειλές κατά τη διάρκεια της παραγωγής IP καθώς και επίσημες μεθόδους επαλήθευσης και αναγνώρισης της ασφάλειας για την αποτροπή τυχόν ανεπιθύμητης διαρροής πληροφοριών. Στο κεφάλαιο 2 ασχολούμαστε με την παραγωγή των IC , στο κεφάλαιο 3 εξετάζουμε τις απειλές για την ασφάλεια των IC κατά τη διάρκεια της παραγωγής, ενώ στο κεφάλαιο 4 εξετάζουμε την επαλήθευση ασφάλειας των 3PIP (πνευματική ιδιοκτησία τρίτων) και τις μεθόδους ταυτοποίησης.

Abstract

The quality of computer hardware is frequently taken for granted by software and system developers. Advancements in semiconductor processing technology and the System-on-chip (SoC) design flow globalization has given rogue intellectual property (IP) providers the chance to introduce malicious circuits (also known as hardware Trojans) into their IPs.

This thesis presents threats during the IP production as well as formal security verification and identification methods to prevent any unwanted information leakage. In chapter 2 we elaborate in IC manufacturing and chapter 3 is dedicated in security threats in IC manufacturing and supply chain while in chapter 4 we dive into formal security verification of 3PIP (third party intellectual property) and identification methods.

Table of contents

Chapter 1
Introduction

The complexity of integrated circuits rises together with the size scaling and the exponential growth of their functionality. The complicated methods needed to design and fabricate today's advanced chips are expensive, time-consuming, complex, and they can only be carried out in cutting-edge fabrication facilities. Over the past two decades, the semiconductor business model has mainly changed to a contract foundry business model (also known as a horizontal business model) due to the rising cost and complexity of foundries and their processes.

The relationship between the designer and the foundry is asymmetrical under the horizontal business model: the designed IP is accessible to manufacturers, who can recreate the ICs with a low overhead due to the ease of access of the masks. However, the specifics of the fabrication process, quantity, and any changes to the original designer's chip blueprint are secret to the design house. The current corporate structure and contractual arrangements are insufficient to fully protect the designer's intellectual property rights. The IP owners pay for the expensive construction of masks based on their designs and also disclose the specifics of their IP. They have faith in the foundry to respect their intellectual property and avoid overproducing ICs. The mask is easily accessible at the foundry and is rather inexpensive.

A set of security protocols called "IC metering" enables design houses to have post-fabrication control over their ICs. In 2001, the phrase "hardware metering" was initially used to describe the first passive technique for identifying each IC's functioning while preserving the same input/output behavior and synthesis flow. Simultaneously, techniques for identifying ICs with

physical unclonable functions (PUFs) have been under development. Since then, numerous newer strategies for offering a design house administration over their designs have been put forward for metering. Note that the attack model, the intended level of protection, and the security assumptions for the IC supply chain all influence the levels of post-fabrication protection and control.

In this thesis, existing counterfeit detection methods will be presented but more specifically those that fall into the formal verification category. Identification techniques will be the second part that will be reviewed as well as thoughts and observations.

Chapter 2
IC manufacturing

Modern electronics, including computers and smartphones, are built around integrated circuits (ICs), which are crucial parts of many different sectors. A single chip of an IC contains thousands or even millions of electrical components. These devices are made using an intricate and highly specialized process known as integrated circuit manufacture. The design of the chip is the first step in the manufacturing of an IC and to design the circuit layout and specify the behavior of the components, IC designers employ specialized software tools. There are multiple crucial steps in the manufacturing of ICs which are presented below.

Design and Specification: Engineers determine the functionality and performance criteria of the integrated circuit during the design and specification stage of the IC manufacturing process. System-level architecture design, circuit design, and verification are steps in this process that make sure the design adheres to the intended standards. To construct a thorough layout of the circuit's parts and connections and to simulate the design's behavior for functional verification, designers employ specialized software tools.

Mask Generation: The creation of the mask comes after the IC design has been completed. In order to do this, the design must be transformed into a collection of photomasks, which are precise templates used in the production of semiconductors. The patterns on the masks specify the precise positioning and forms of the different components on the IC. The masks are created using sophisticated computer-aided design (CAD) tools while taking into account the specific requirements and constraints of the production process.

Wafer Fabrication: The IC design is physically translated onto a silicon wafer during wafer manufacture. Wafer cleaning, wafer doping, and the deposition of numerous layers of materials like silicon dioxide and polysilicon are just a few of the sub-steps that make up this process. The photomasks are used in the crucial process of photolithography, which transfers the design patterns onto the wafer's surface. To construct the desired circuit components, such as transistors, interconnects, and metal layers, the wafer is subjected to a number of etching, deposition, and implantation processes. In order to construct the intricate integrated circuit, each layer is precisely aligned and patterned.

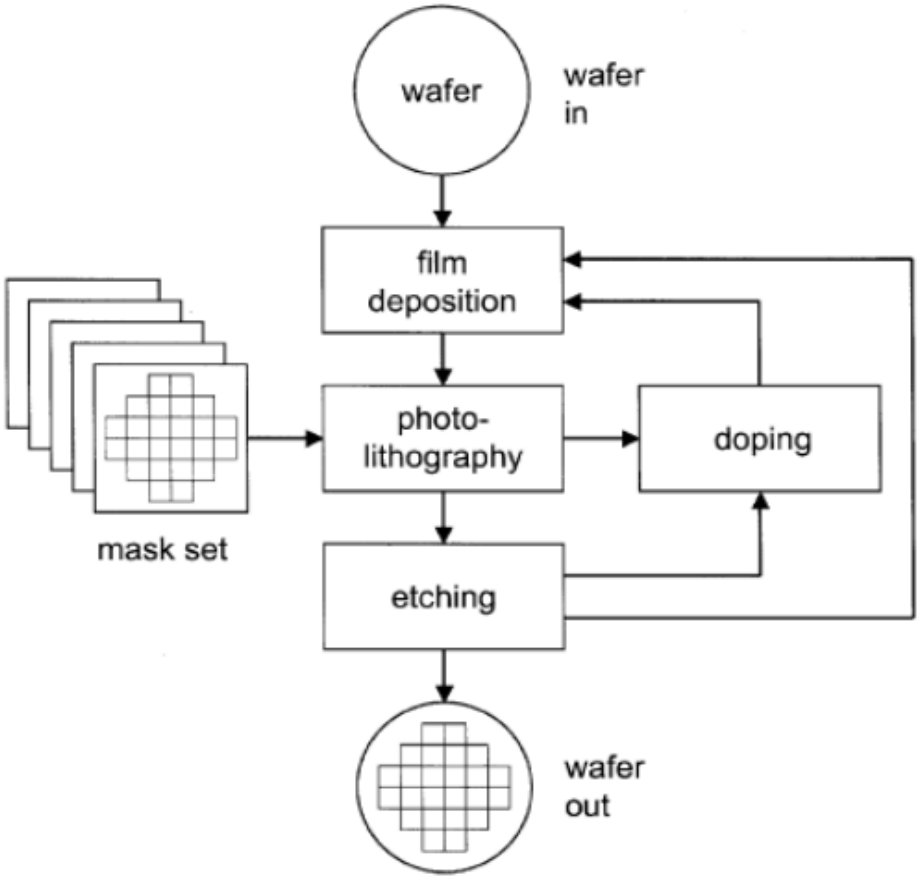

Fig 1. Flow diagram of an IC fabrication process [10]

Packaging and Testing: The individual ICs are separated when wafer manufacture is finished and go through packaging. Wire bonding or flip-chip bonding is used to provide electrical connections when packaging encloses the IC in a protective shell. The packed ICs are then put through a thorough testing process to assure their performance and functioning. To ensure that the ICs meet the criteria, various tests are carried out, including electrical, functional, and temperature testing. The good ICs are identified, labeled, and prepared for distribution while the flawed or non-functional ones are rejected.

Quality Assurance and Reliability: To ensure the long-term performance and dependability of the ICs, quality assurance and reliability measures are implemented in this final step. To evaluate the ICs' robustness under various operating situations, several reliability tests are included, such as burn-in tests, environmental stress testing, and accelerated aging tests. In order to monitor and validate the manufacturing processes and guarantee the consistent and dependable production of high-quality ICs, strict quality control measures are also put in place.

Chapter 3
Security threats specific to IC

ICs are manufactured using a variety of processes, including design, fabrication, testing, and packaging. Different security threats that could endanger the integrity and confidentiality of the IC design exist at each level. The IC supply chain includes distributors, design houses, foundries, packaging and testing organizations, and other stakeholders. The unique security threats that each participant poses may compromise the IC design's integrity and secrecy. This section will discuss the most common security risks that impact IC manufacturing/supply chain and their effects.

The following are the main dangers to the security features of ICs:

• Reverse engineering to extract IP or find sensitive information (like cryptographic keys) stored in on-chip memory

• Tampering to undermine IC operation or introduce malicious functionality, such as Trojan horses or kill switches (threat to integrity and dependability)

• Counterfeiting (a danger to authenticity and frequently owing to the poor quality and dependability of counterfeits)

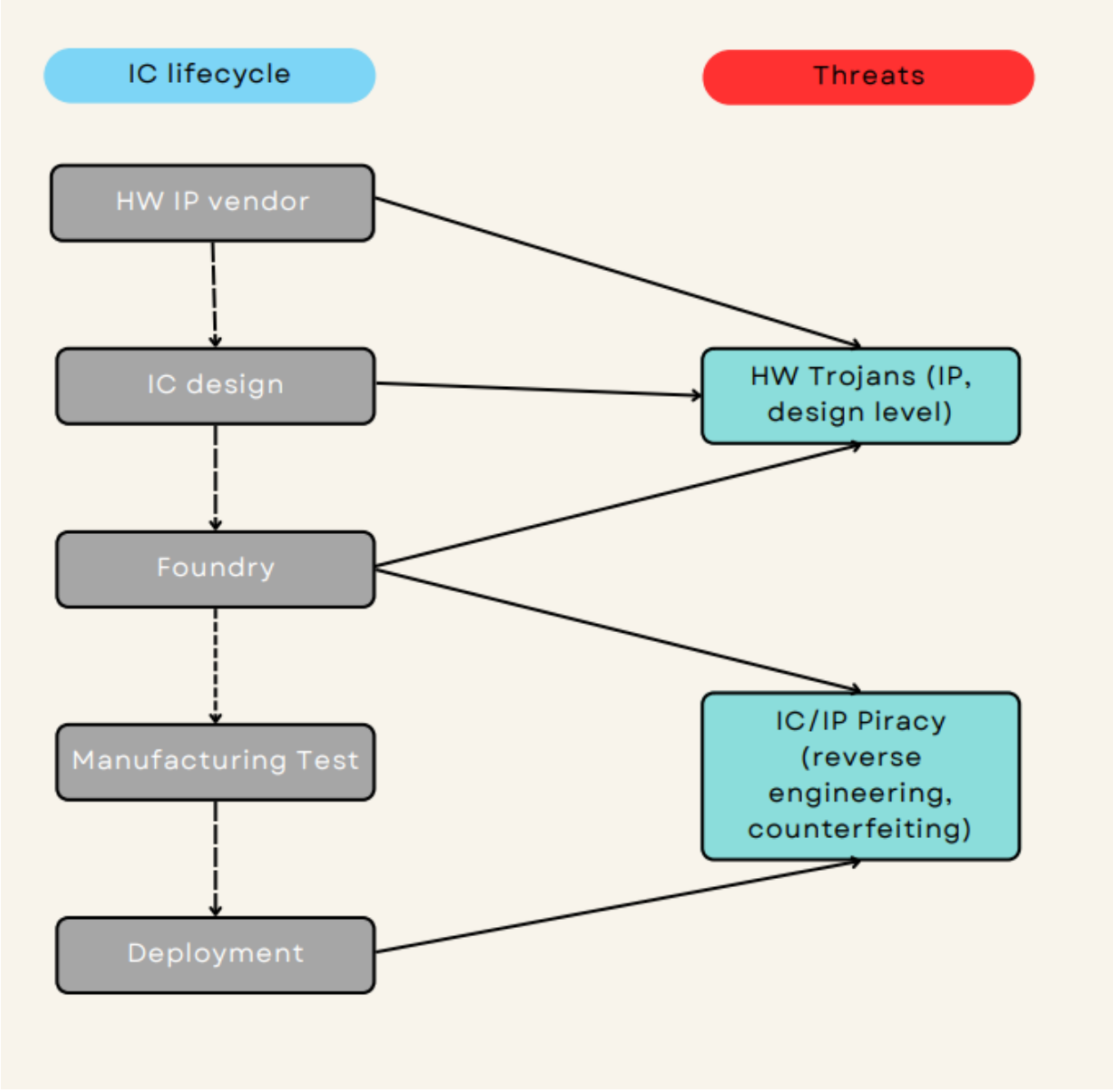

Fig 2. Steps of an IC becoming from an idea to reality in pair with the threats along the way.

3.1 Reverse Engineering

Physical reverse-engineering attacks, which can be invasive or non-invasive, are used to gather details regarding the IC's functionality. By depackaging, either artificially or completely removing the IC's packaging, invasive attacks, also known as destructive physical inspection attacks, are carried out. This can be done by using acids, solvents, or other chemicals, physical abrasion methods including grinding, chemical or mechanical polishing, or laser cutting the packing material to vaporize it. The circuitry can be scanned once the IC has been depackaged as every circuit layer is gradually exposed via grinding. Circuit access also makes it possible to launch "reconnaissance" attacks, such as those that involve tracking down targets for electromagnetic attacks or reverse-engineering the IC's circuits. The metal tracks of the IC can also be exposed to measure voltages and signals or to actively introduce signals. The IC's circuitry can be modified, or inactive self-test circuitry can be activated again using a focused ion beam (FIB). In order to see the fine metal tracks, it can also be used to make tiny holes in the insulating layer of the IC [1].

In non-invasive assaults, physical characteristics or signals connected to physical processes that take place while the IC is operating are observed. To learn more about the state of the IC and the data it processes, these physical signals can be studied. Device timing and clock rates, electrical voltage levels and power consumption (simple and differential), temperature levels, electromagnetic (EM) radiation, acoustics, and light emission can all be used to derive signals. An attacker searches for anomalies like power usage alternations or hiccups in clock frequency. They can also use the IC's traceable signals to purposefully introduce problems into the device's operation. In order to determine whether cryptographic devices' timing, supply voltage, and electromagnetic side channels could be used to find cryptographic keys or identify covert data leakage, cryptographers have long studied them. These studies are known as side-channel attacks. In more recent years, scientists have looked into how to leverage side-channel analysis, such as temperature, performance analyses, timing channel and gate to find kill switches and hardware Trojans [1].

Physical attacks are still uncommon despite being challenging to stop since they involve so much money, skill, and patience and are so technologically advanced. For instance, while it is theoretically possible to modify an IC's wiring and get around security measures, doing so requires expensive equipment and extensive knowledge, especially when aiming for a modern IC that has been manufactured with nanoscale feature sizes. FPGAs are an exception to the "difficulty" criterion because they are particularly prone to having their intellectual property (IP) duplicated, just like software programs can. Because their memory is volatile or needs to be reinitialized every time power is applied, conventional SRAM-based FPGAs are particularly vulnerable. A different external bitstream needs to be put into the FPGA for each re-initialization. The IP of the FPGA can be readily and non-intrusively duplicated using that external bitstream as a conduit [1].

The non-volatile memory is housed on the FPGA chip itself, making it the most secure FPGA to have a single chip. The data in the on-chip memory is encrypted using the FPGA's strong encryption capabilities, in addition to IP and bitstreams used for programming. The bitstream identifiers and encryption keys are also stored in the non-volatile memory registers. The IP and data stored in FPGAs that are vulnerable to physical "sand-and-scan" reverse engineering or data extraction attacks are also protected by encryption [1].

3.2 Tampering

Since it is practically impossible to tamper with created chips in a way that is fine-grained enough to alter the logic of the hardware without simply destroying the hardware, tampering to change the functionality of an IC other than an FPGA is always done to the design of the IC. FPGAs provide a larger risk of post-manufacture manipulation since their system programming can only be changed safely in the presence of secure IC programming and data protections that limit access to the FPGA's intellectual property and the data kept in its on-chip memory [2].

A growing number of ICs now feature built-in defenses against physical attacks due mostly to IC manufacturers' worries about physical tampering to obtain IP. For instance, the IBM 4758 co-processor "wraps" its hardware in a "shell" that detects and reacts to tampering. Others encrypt their IPs so that even if the IC is physically attacked, the attacker cannot decode the IP [3]. Additionally, methods for hiding IC logic have been developed and are being improved in order to better defend ICs against intellectual property reverse engineering [4].

The Department of Defense's (DoD) Anti Tamper (AT) research project, which aims to create technologies that can stop reverse engineering and the extraction of intellectual property from ICs used in sensitive DoD systems and applications, has been producing a number of intriguing anti-tamper techniques [5]. One such device is IC metering, which offers a collection of security protocols intended to let an IC design company keep control over an IC even after it has been manufactured. Such control may be active, such as incorporating the IC with the capacity to automatically disable itself at run-time if any hint of tampering is discovered, or it may be passive, such as restricting the production of ICs and the characteristics they exhibit [5].

The deliberate corruption of hardware typically takes place during its design, implementation, or manufacturing—well before the malicious circuitry is activated—much like time bombs and logic bombs in software. But unlike software, sabotaged ICs, with the exception of FPGAs, cannot be fixed, thus they continue to pose a hazard. Well-crafted IC-level vulnerabilities or malicious insertions would probably need to have the affected hardware physically replaced.

The difficulty of replacing hardware, especially in highly embedded systems, would guarantee that corrupted ICs continue to function even after the Trojan or vulnerability is identified.

Additionally, since the IC is in the computer system's lowest layer, "unwanted" logic at that level can offer a way to get around, override, or manipulate of any software that runs in the upper levels, enabling for the creation of sophisticated and covert attacks that are intended to bypass defenses that use software as their basis. Even low-level software like virtual machines and kernel-level processes may only deter attacks that originate from within the processor on which the software is loaded, not prevent them. This is true for any protection that merely consists of numerous layers of security provided by software [6].

Attackers could insert a specific sequence of bytes into the IC, for instance, to activate embedded malicious circuits, allowing leaks of highly sensitive data or cryptokeys, stopping the processor at crucial or arbitrary processing times, searching for electromagnetic signals that serve as the external cues for processor shut downs, or facilitating reverse engineering of the IC design. More advanced hardware Trojan logic has been developed that gives attackers the ability to elevate privileges, disable access control checks, and carry out arbitrary instructions, giving them a path to seize control of the computer and a base from which to launch further system-level attacks. A packet transmitted from a specified network address, or a key encoded as a sequence of requests to various ports, for instance, might be used as the catalyst for the Trojan to "reset" the firewall, enabling full unrestricted access to the network. [7].

However, many hardware attacks could be mistakenly attributed to having issues with the design or various manufacturing problems instead of intentional reasoning since their effects often seem exactly like "normal" hardware failures.

3.3 Counterfeit IC

Producing nearly identical copies of authentic products or product data (such as certificates of authenticity) is known as counterfeiting. These copies are so similar to the originals that a typical user, reseller, tester, or other non-expert observer might mistake them for the real thing. Unreliable IC fabricators, especially those located overseas, make use of the capacity to reproduce an IC's intellectual property (IP; remember, this is its design) for use in counterfeiting or "overbuilding." Overbuilding, also known as "run-in fraud," is a type of intellectual property theft and IC counterfeiting when a subcontractor to an IC manufacturer steals the IP from the ICs they are subcontracted to create and then inserts that IP into less expensive ICs that are bought on the open market. The firm subsequently competes directly with the original equipment manufacturer by selling the ICs holding the stolen intellectual property [23].

But when it comes to ICs, the majority of fakes aren't copies; instead, they're real ICs that have been tampered with or inaccurately portrayed. They are frequently recovered from obsolete computer boards or electronic devices, resurfaced, and given a newer revision number. They are then delivered with documentation that exaggerates their true performance and mechanical characteristics compared to the real ICs that they are imitating [23].

The majority of IC original equipment manufacturers (OEMs) have made significant investments in creating safeguards to protect their in-chip IP since the potential of counterfeiting and overbuilding is so substantial. These techniques include fingerprinting, obfuscation, watermarking, and encryption. The majority of OEMs also provide security features for uploading the programming bitstreams needed to reprogram FPGAs, like bitstream encryption and authentication. A FPGA will only accept programming bitstreams whose integrity can be verified using message authentication codes thanks to bitstream authentication. Some OEMs also offer authenticated remote hardware update channels to their clients, preventing the upload

of subverted update bitstreams with harmful design logic. When an attempt is made to reverse engineer the IC, the hardware's description or structure is altered in a way that purposefully hides its functioning. Hardware IP watermarking involves hiding the identity of the IP holder in the IC's description, where it can later be found and used to confirm the IC's provenance [8].

The Physical Unclonable Function (PUF), which has been developed by the research community, is one authentication method for ICs. PUFs, which have distinct physical traits, appear as process variances in every IC in a run that is made from the same silicon mask. A one-way challenge-and-response function can be used to authenticate an IC's PUF-based identifier. To do so, the IC must correctly locate the output from one or more challenge inputs. This output should be specific to the IC because of the uniqueness of its PUF's process variation, and it serves as the foundation for the authentication of the IC's PUF-based identifier [9].

The price of an IC, however, may ultimately be a much clearer indicator that it is a fake. The majority of imitation goods, whether they be wallets, medications, or processors, are sold for significantly less than the original.

Chapter 4
Formal verification and identification

Formal verification is a strategy to guarantee that every safety-critical design element is thoroughly checked for accuracy [11]. The qualities provided in temporal logic and its variants can be used to specify quality requirements [12]. The concept of time in linear time temporal logic (LTTL) is that of a linearly ordered set, which can be conceptualized as a potential series of states.

The practice of examining a design for the truthfulness of properties specified in temporal logic is known as model checking. A model checker creates a Boolean satisfiability (or SAT for short) formulation for validating/invalidating a property using the Verilog code and the property written as a Verilog assertion. A SAT engine processes this SAT formulation and looks for an input assignment that violates the property [13].

Bounded model checking (BMC). Designers are aware of the maximum number of steps (clock cycles) that a property can withstand in practice. A property is found to hold in BMC to at least a finite number of state transitions. A SAT engine is provided with the Boolean formula to validate or invalidate the target property, and if a satisfying assignment is seen within T clock cycles, the assignment acts as a witness against the target property [14]. Criteria to identify Trojans that tamper with crucial data are created with a bounded model checker to ensure that the target design satisfies these criteria.

4.1 Formal verification methods

As suggested in [15], a new IP acquisition and delivery protocol can assist IP customers in swiftly determining the reliability of 3PIP they bought from IP providers. Our suggested relationship between IP sellers and customers is depicted in Fig 3. and it shows a predefined set of security requirements that the IP must meet can be agreed upon by a SoC integrator and a 3PIP vendor.

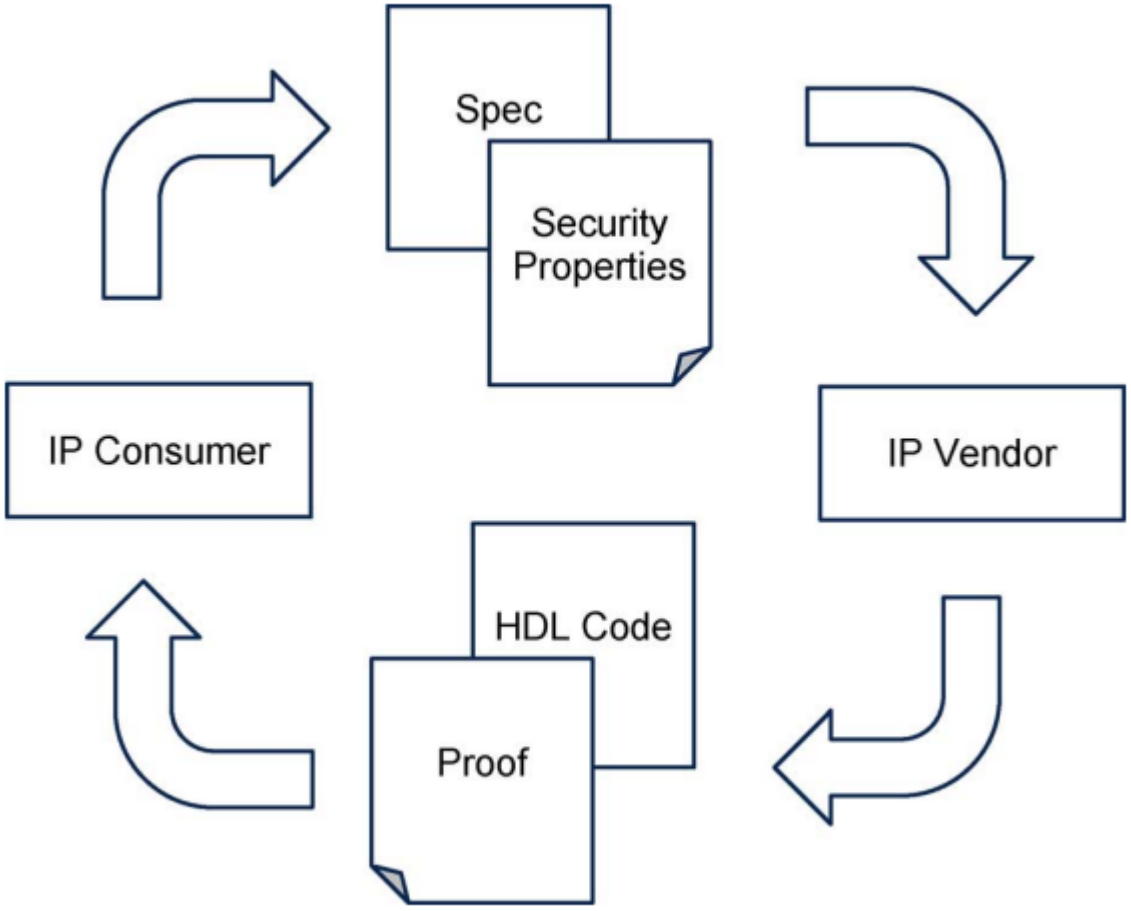

Fig 3. IP acquisition and delivery protocol [15].

The 3PIP's properties can be checked by the SoC integrator. One translates the desired design into a proof checking format, like Coq, to see if it upholds the predetermined criteria. Coq is a formal proof management system and an interactive theorem prover. It provides a language and environment for writing formal specifications and verifying the correctness of mathematical proofs and software programs using formal logic and type theory. The method has been proven in [16] and [17] to be effective in data leakage detection and malicious tempering respectively. Restrictions of this method are:

The trustworthiness of the VHDL/Verilog representation is not inherently linked to the Coq representation. The VHDL/Verilog code may still contain errors, bugs, or vulnerabilities that

were not identified during the translation process or the subsequent formal verification in Coq. Furthermore, the VHDL/Verilog representation can be subject to implementation-specific issues, such as synthesis optimizations, timing constraints, and technology-specific considerations, which are not captured in the Coq representation.

By converting VHDL/Verilog designs into Coq format, it becomes possible to formally reason about the correctness, security, and other properties of the hardware design using Coq's powerful proof techniques. This conversion enables the application of formal verification methodologies to ensure that the hardware design meets its specifications and is free from potential security vulnerabilities or flaws. But, with this technique there is no automation in converting VHDL/Verilog to Coq [17].

Additionally, while the design might agree with the predefined properties, if there are vulnerabilities, those cannot be checked.

Threat model: The threat model that will be used by both paradigms will be the one used in [18]-[19]. The attacker is a 3PIP vendor or a rogue element of a 3PIP vendor company. The attacker aims to compromise the SoC's security that uses his IP. He inserts hardware Trojans into the IP to tamper with important data. He solely uses Trojans with "digital" trigger and payload features. The physical properties of the SoC are established by the design-synthesis restrictions, and the 3PIP manufacturer has no control over the design constraints that the SoC integrator imposes on the SoC. As a result, he cannot create a Trojan that depends on these properties. The Trojans are being evaluated that they don't employ non-volatile components because a designer can designate "non-volatile" components in a 3PIP as he needs to create them.

The defender is the SoC integrator. His goal is to find any Trojans present in the 3PIP. The defender is assumed to have access to the 3PIP's RTL/gate-level netlist and can therefore confirm its functionality. Additionally, he is familiar with the 3PIP's input and output ports according to the specification.

Protocol to check the reliability of an IP. Just as the procedure described in [15]–[17], a set of security features for the design are decided upon by the SoC integrator and the IP provider. For instance, one attribute might verify the acceptable methods of updating a processor's stack pointer. Functional verification is performed by a SoC integrator. Attackers within the IP design house have the ability to introduce Trojans that corrupt crucial data while meeting the agreed-upon security parameters and passing functional testing. The SoC integrator has two tasks: (1) determine whether the design complies with the agreed-upon security properties, and (2) determine whether the design contains Trojans that corrupt data without infringing upon these properties [15]-[17].

4.2 Identifying data corruption

Formally, a data corrupting Trojan is described as,

$$\exists \; i_{trigger} \in I, i_{trigger} \notin \mathbb{V} \ni D \models (R_{t-1} \neq R_t) \qquad (1)$$

where I is the collection of potential input patterns spanning all clock cycles [1, t], Rt is the value of the register R at clock cycle t. The design is D, and the set of legal ways to alter R is V. R becomes corrupted when itrigger is applied. The important register is the one that contains the crucial information.

This crucial register could be a stack pointer in a processor, a destination address register in a router, or a key register in a cryptographic architecture. Below, various paradigms are being presented of data corruption identification via formal verification.

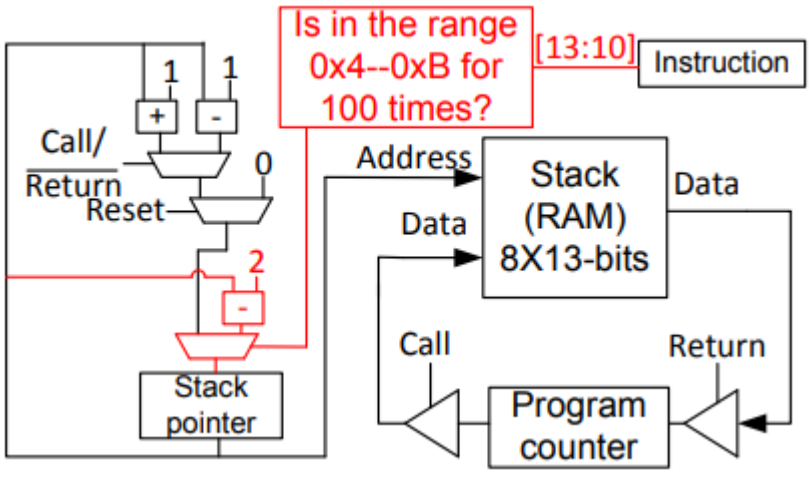

Fig 4. A RISC processor's stack pointer was tampered with by a Trojan [20]. The CALL instruction increases the stack pointer's value by 1, the RET instruction decreases the value by 1, and the RESET instruction sets the stack pointer's value to zero. When the bits [13:10] of the instruction register are in the range 0x4-0xB for 100 times, the Trojan is activated. The Trojan reduces the value of the stack pointer by 2. The red components are those of a trojan.

Example 1: When bits 13–10 of the instruction are in the range 0x4-0xB for 100 times, as shown in Figure 4, the stack pointer of a RISC processor becomes corrupted. The Trojan decrements the stack pointer by two when it is activated, opening the door for control-flow attacks.

A Trojan can be (i) always on (i.e., no trigger), (ii) only on by current inputs, (iii) only activated after a certain number of clock cycles, and (iv) only activated by inputs arriving over a period of several clock cycles [2, 3]. The types (i) and (ii) of Trojans are handled by FANCI and VeriTrust. To combat these, DeTrust created Trojans of types (iii) and (iv). All varieties of data-corrupting Trojans can be found using this method [18].

Example 2: In the RISC processor's stack pointer in Figure 4, BMC creates a counterexample with 100 ADD instructions to test for stack pointer corruption using the property. Keep in mind that in a Trojan-free design, an ADD instruction shouldn't impact the stack pointer.

Partial corruption: as it checks for individual bits of R, the property can find this Trojan even if an attacker updates a subset of the bits in R [18].

Multiple-cycle triggers: It considers whether a trigger occurs over a single clock cycle or over several [18].

Example 3: The Trojan in the Trust-hub's AES-T800 [20] taints the eight least important bits of the secret key. To activate the Trojan, one must apply four pre-selected plaintexts in a row. If these plaintexts do not come in order, the Trojan will not be launched. BMC uses to create a counterexample for the no-data-corruption property, contains the four plaintexts that launch the Trojan.

Example 4: If the design is unraveled for 400 clock cycles, BMC has the capacity to identify a list of laws that launch the Trojan in Figure 1. However, if the design is unrolled for fewer than 400 clock cycles, BMC does not provide a counterexample. The maximum number of clock cycles must thus be verified.

Automatic test pattern generation (ATPG) is a technique that may be used to ensure dependability across a large clock cycles number [25]. The target circuit's monitor circuit is used as the model for the property [26]. The output of this monitor circuit can be used to test for a stuck-at-1 fault in order to cause the ATPG to generate an input pattern that violates this property. If an ATPG generates a test pattern (counterexample), the property is violated. If the flaw is not found, the attribute is true. If an ATPG returns as untestable, the design's dependability is not guaranteed. The monitor circuit is not built in silicon and is simply required for validation.

In table 1[10] we have the results of DeTrust exploiting FANCI's and VeriTrust's [21] limitations to design trojans that can bypass them. Also, the results of bounded model checking and automatic test pattern generation.

Note that the technique by [18] based on BMC and ATPG were able to detect all Trojans.

Trojan			FANCI	VeriTrust	BMC				ATPG			
Name	Trigger condition	Payload	Detected?	Detected?	Detected?	Time (s)	Memory (GB)	Max.# of clk cycles	Detected?	Time (s)	Memory (GB)	Max.# of clk cycles
MC8051-T400	Instruction = (1) MOV A, Data (2) MOVX A, @R1 (3) MOVX A, @DPTR (4) MOVX @R1, A	Prevents interrupt	No	No	Yes	0.01	0.01	140	Yes	0.001	0.01	480
MC8051-T700	MOV A, Data	Modifies the data to 0x00	No	No	Yes	0.01	0.09	160	Yes	0.001	0.01	450
MC8051-T800	Input data of UART =0xFF	Decrements stack pointer by two	No	No	Yes	0.01	0.1	160	Yes	0.001	0.01	430
RISC-T100	After 100 instructions whose 4 MSBs are in the range 0x4–0xB	Increments program counter by two	No	No	Yes	73.86	1.2	110	Yes	8.57	0.2	360
RISC-T300		Modifies the data written to memory	No	No	Yes	75.63	3.2	120	Yes	0.56	0.19	360
RISC-T400		Modifies the data address to 0x00	No	No	Yes	74.67	1.8	120	Yes	0.58	0.19	360
AES-T700	Plaintext = 128'h00112233445566 778899aabbccddeeff	Modifies LSB 8-bits of key register	No	No	Yes	40.6	2.01	50	Yes	0.22	0.74	150
AES-T800	Plaintext = (1) 128'h3243f6a8885a3 08d313198a2e0370734 (2) 128'h0011223344556 6778899aabbccddeeff (3) 128'h0 (4) 128'h1	Modifies key register	No	No	Yes	40.6	2.01	70	Yes	0.41	0.77	160
AES-T1200	After 2^{128} clock cycles	Modifies key register	No	No	N/A	N/A	N/A	160	N/A	N/A	N/A	160

Table 1[18]: Detecting Trojans from the Trust-Hub and were structured from DeTrust [21]. The clock cycle in which the trigger occurs is indicated by the number in parentheses in the trigger condition column. No counterexample was discovered to break the property, as shown by "N/A". To determine the greatest number of clock cycles for which a design can be unrolled, [18] ran the tools for 100 seconds.

4.3 Identifying information leakage

Similarly [19], in information leakage detection, there were ups and downs using (2)

$$\exists i \in I \ni D \models (s == o) \qquad (2)$$

so, considering the Trojan AES-T100 from Trust-Hub [20] and using the refined (3),

$$\exists i \in I, s_x \in S_{N-1} \ni D \models (s_0 == o) \qquad (3)$$

the aim now is to check for the secret key's subset leakage. Only the eight least important bits of the secret key are revealed by this Trojan. The Trojan cannot be detected by the initial property since its most important 120 bits are not exposed. There are 2^128 possible subsets if one updates the initial property to check for the leaking of a subset of the key bits, making it computationally impossible to check for every conceivable subset.

Refinement 1 (equation 2) searches for a Trojan that exposes the secret key's target bit (bit s0). The BMC will assign values to the other bits of the secret key and the inputs while examining this attribute to see if it is satisfied. The property discovers a Trojan if there is an assignment. The number of attributes examined by the defender for an N-bit key decreases from 2N to N by focusing on the individual key bits rather than the subset of key bits. In addition, the target bit s0 can only be given the values logic 0 or 1. Therefore, rather than checking for 2 N potential values, one should do it for 2 N.

Now considering there is a multiple clock cycle trigger, there are Trojans with trigger vectors that span several clock cycles. Such Trojans are undetectable by the first refinement and property. That is why in AES-T800 again from the Trust-hub [20] as shown in the figure [5] below,

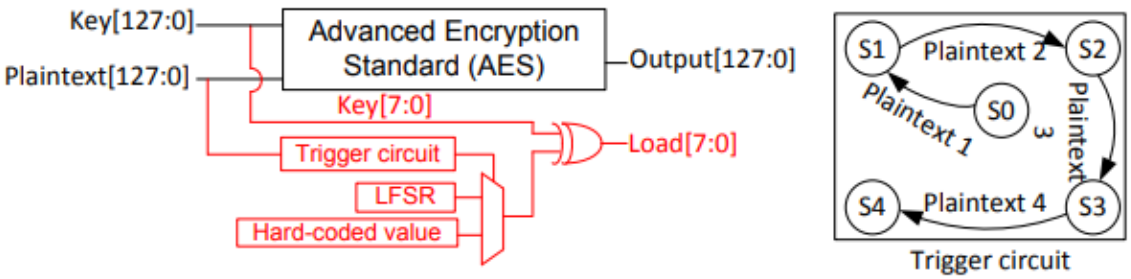

Fig 5. [19] Using a hardware Trojan in AES design. Through the load output, the Trojan exposes the least important 8 bits of the secret key. To activate the Trojan, one must use one of four pre-selected plaintexts (1 through 4). The key is XORed with a predetermined constant once the Trojan has been activated. Otherwise, a linear feedback shift register (LFSR)'s data is XORed with the key.

Through the load output, the Trojan spills the eight least important bits of the secret key. To activate the Trojan, four pre-selected plaintexts (1 through 4) must be applied in order. If these plaintexts don't come in order, the Trojan won't be activated. The key is XORed with a predetermined constant once the Trojan has been activated. Otherwise, a linear feedback shift register (LFSR)'s data is XORed with the key.

One must ascertain the assignments to the input over a number of clock cycles in order to detect such Trojans. BMC is used to implement a second refinement. BMC attempts to identify a set

of input assignments over those clock cycles that violates this requirement by unrolling the design for a number of clock cycles.

Numerous examples take place in [19] and the results appear in Table 2.

Benchmark characteristics			Detection capability						
Name	Key bits leaked	Trigger condition	Property				Detected?	Memory (GB)	Time (s)
			Initial	Refinement 1	Refinement 2	Final			
AES-T100	7-0	Always on	No	Yes	Yes	Yes	Yes	3.91	96.1
AES-T200	7-0	Always on	No	Yes	Yes	Yes	Yes	3.92	94.23
AES-T600	127-0	Plaintext =128'hf	No	Yes	Yes	Yes	Yes	3.96	96.8
AES-T700	7-0	Plaintext = 128'h001122334455 66778899aabbccddeeff	No	Yes	Yes	Yes	Yes	3.94	96.33
AES-T800	7-0	Plaintext = (1) 128'h3243f6a8885a3 08d313198a2e0370734 (2) 128'h0011223344556 6778899aabbccddeeff (3) 128'h0 (4) 128'h1	No	No	Yes	Yes	Yes	3.95	96.86
AES-T900	7-0	At clock cycle 2^{128}	No	Yes	Yes	Yes	Yes	3.93	96.12
AES-T1000	7-0	Plaintext = 128'h00112233445566 778899aabbccddeeff	No	Yes	Yes	Yes	Yes	3.94	96.55
AES-T1100	7-0	Plaintext = plain texts consecutively (1) 128'h3243f6a8885a3 08d313198a2e0370734 (2) 128'h0011223344556 6778899aabbccddeeff (3) 128'h0 (4) 128'h1	No	No	Yes	Yes	Yes	3.95	96.57
AES-T1200	7-0	At clock cycle 2^{128}	No	Yes	Yes	Yes	Yes	3.94	96.87
AES-T2000	127-0	Plaintext = (1) 128'h3243f6a8885a3 08d313198a2e0370734 (2) 128'h0011223344556 6778899aabbccddeeff (3) 128'h0 (4) 128'h1	No	No	Yes	Yes	Yes	3.98	94.92
RSA-T100	31-0	Plaintext = 32'h44444444	No	Yes	Yes	Yes	Yes	0.27	2.31
RSA-T300	31-0	Every alternate clock cycle	No	No	Yes	Yes	Yes	0.28	2.48

Table 2: [19] Trust-Hub benchmark suite's features have the capacity to detect Trojans [20]. The number of clock cycles is indicated in parentheses in the trigger condition column.

Chapter 5
Observations

5.1 Discussing the methods

The Trust-hub benchmark suite designs to include pseudo-critical and bypass registers in order to evaluate the effectiveness of the strategies in doing so. DeTrust Trojans were utilized[20]. [18] used the tools for this experiment for 100 seconds.

When a register is simple to manipulate, it is simpler to check its pseudo-critical properties, and when a register is simple to observe, it is simpler to check its bypass properties. The AES critical register, also known as the key register, is located closer to the inputs. As a result, controlling something is easier than observing it.

As a result, more clock cycles must be unrolled to check for pseudo-critical registers than for bypass registers. On the other hand, in MC8051 and RISC, the critical registers are located closer to the outputs. As a result, they are simpler to observe than to govern. Therefore, compared to bypass registers, fewer clock cycles are unrolled to check for pseudo-critical registers.

The amount of clock cycles unrolled by ATPG is 2.5 times greater than that by BMC, similar to Table 1. All designs were unrolled for more than 1000 clock cycles when given enough time (30 minutes).

In [19], a property to identify Trojans that leak information is also proposed, provided that the information leak occurs within the maximum number of clock cycles for which the design is unrolled. For clock cycles above this number, no security guarantees are provided. As a result, the throughput is decreased when the number of clock cycles reaches this limit and the design must be reset. To achieve a throughput reduction of less than 1%, one can unroll the designs for more than one hundred clock cycles. However, utilizing an automatic test pattern generator (ATPG) in place of a BMC to check for the attribute can increase the number of clock cycles for which the design is examined. The reason for this is that an ATPG uses less memory than a BMC. Similar characteristics can be developed to identify Trojans that alter a design's functionality and Trojans that damage registers that store important data.

5.2 Conclusion

In conclusion, formal verification stands as a critical and powerful tool for ensuring the correctness, security, and reliability of complex hardware and software systems. Through the rigorous application of mathematical reasoning and formal methods, formal verification enables the detection and elimination of design flaws, bugs, and vulnerabilities early in the development process. It provides a systematic and exhaustive exploration of all possible states and behaviors of the system, ensuring that it meets its specified properties and requirements. Formal verification offers a level of assurance that traditional testing methods cannot provide, enabling designers to have confidence in the correctness and trustworthiness of their systems.

However, the adoption and utilization of formal verification techniques do come with challenges. The computational complexity of formal methods, the need for specialized expertise, and the limitations of modeling and abstraction can pose obstacles to their widespread application. Nonetheless, ongoing research and advancements in formal verification methodologies continue to address these challenges, making formal verification increasingly accessible and applicable to real-world systems.

In future research, the focus should be on further enhancing the scalability, automation, and usability of formal verification techniques. Additionally, exploring the integration of formal verification with other verification and validation methodologies can lead to more comprehensive and efficient approaches for ensuring system correctness and security. As technology advances and systems become more intricate, formal verification will play a vital role in guaranteeing the integrity, security, and reliability of the hardware and software systems that underpin critical applications in various domains.

References

1. Mertz, Michael. “Secure in-system programming for FPGAs”. 2005.
2. Landis, Dave. “Programmable Logic and Application Specific Integrated Circuits”
3. Kastner, Ryan, and Ted Huffmire. “Threats and Challenges in Reconfigurable Hardware Security”. Proceedings of the 2008 International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA ’2008). Las Vegas, NV, 14-17 July 2008.
4. Rajendran, Jeyavijayan, et al. "Security analysis of logic obfuscation." *Proceedings of the 49th annual design automation conference*. 2012.
5. DoD Anti-Tamper Program Website.
6. Koushanfar, Farinaz. “Integrated Circuits Metering for Piracy Protection and Digital Rights Management: An Overview”. Proceedings of the 21st Association of Computing Machinery (ACM) Great Lakes Symposium on Very Large-Scale Integration (GLSVLSI ’11). Lausanne, Switzerland, 2-4 May 2011. http://aceslab.org/sites/default/files/GLS-VLSI.pdf
7. Abramovici, Miron, and Paul Bradley. Integrated Circuit Security—New Threats and Solutions. Proceedings of the Fifth Annual Cyber Security and Information Intelligence Research Workshop (CSIIRW ’09). Oak Ridge, TN, 13-15 April 2009.
8. Huffmire, Ted, et al. “Managing Security in FPGA-Based Embedded Systems. IEEE Design and Test of Computers (Volume 25 Issue 8, November/December 2008).
9. Helinski, Ryan. “Physical Unclonable Functions”. 27 October 2009.
10. Hierlemann, A., et al., Microfabrication techniques for chemical/biosensors. Proceedings of the IEEE, 2003. 91(6): p. 839-863.
11. J. Woodcock, P. G. Larsen, J. Bicarregui, and J. Fitzgerald,“Formal Methods: Practice and Experience,” ACM Computing Surveys, vol. 41, no. 4, pp. 19:1–19:36, 2009.
12. A. Pnueli, “The temporal semantics of concurrent programs,” Semantics of Concurrent Computation, vol. 70, pp. 1–20, 1979.
13. “Cadence” http://www.cadence.com/products/fv/pages/default.aspx , 2005.
14. A. Biere, A. Cimatti, E. Clarke, and Y. Zhu, “Symbolic Model Checking without BDDs,” Tools and Algorithms for the Construction and Analysis of Systems, vol. 1579, pp. 193–207, 1999.
15. E. Love, Y. Jin, and Y. Makris, “Proof-Carrying Hardware Intellectual Property: A Pathway to Trusted Module Acquisition,” IEEE Trans. on Information Forensics and Security, vol. 7, no. 1, pp. 25–40, 2012.
16. Y. Jin and Y. Makris, “Proof carrying-based information flow tracking for data secrecy protection and hardware trust,” IEEE VLSI Test Symposium, pp. 252–257, 2012.
17. ——, “A proof-carrying based framework for trusted microprocessor IP,” IEEE/ACM International Conference on Computer-Aided Design, pp. 824–829, 2013.
18. J.Rajendran, V. Vedula, R. Karri, “Detecting Malicious Modifications of Data in Third-Party Intellectual Property Cores”, 2015.

19. J.Rajendran, A.Dhandayuthapany, V.Vedula, R. Karri,“Formal Security Verification of Third Party Intellectual Property Cores For Information Leakage”, 2016.
20. M. Tehranipoor, R. Karri, F. Koushanfar, and M. Potkonjak, “Trusthub,” http:// trust-hub.org
21. J. Zhang, F. Yuan, and Q. Xu, “DeTrust: Defeating Hardware Trust Verification with Stealthy Implicitly-Triggered Hardware Trojans,” ACM Conference on Computer and Communications Security, pp. 153–166, 2014.
22. El Massad, Mohamed, Siddharth Garg, and Mahesh V. Tripunitara. "Integrated circuit (IC) decamouflaging: Reverse engineering camouflaged ICs within minutes., 2015.
23. Goertzel, Karen. Integrated circuit security threats and hardware assurance countermeasures. 26. 33-38, (2013).