Chapter 2
Introduction to Hardware Trojans

Jason Vosatka

2.1 Overview of Hardware Trojans

Hardware Trojans are malicious modifications to the intended functionality of a hardware circuit [9, 17, 36, 37]. These modifications (i.e., tamperings) are undesired and unknown to the hardware designer and can have devastating effects on the electronic system. Trojans have three key characteristics: malicious intention, evasion of detection, and rarity of activation [6]. The intent of a Trojan is always the same: perform an unintended action to compromise the confidentiality, integrity, or authentication of the underlying hardware.

This compromise may be in the form of a shortened operational lifetime of the hardware (e.g., 5 years instead of 20 years) or complete failure of the system upon the Trojan's activation. It may allow an attacker to gain unauthorized access into the hardware (i.e., remote access through a backdoor) or lead to leakage of information (e.g., cryptographic keys for secure data communication). Hardware Trojans may manifest from software Trojans inside of pirated software tool suites during the synthesis portion of the design flow or be inserted as a result of collusion between multiple parties at different stages of the hardware's life cycle [45]. Trojans can also be designed with the sole intention to damage or destroy the brand reputation of a company, which may result in bankruptcy of the company and a competitive advantage for the adversary.

J. Vosatka (✉)
Florida Institute for Cybersecurity Research (FICS Research), Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, 32611, USA
e-mail: jvosatka@ufl.edu

S. Bhunia, M.M. Tehranipoor (eds.), *The Hardware Trojan War*,
https://doi.org/10.1007/978-3-319-68511-3_2

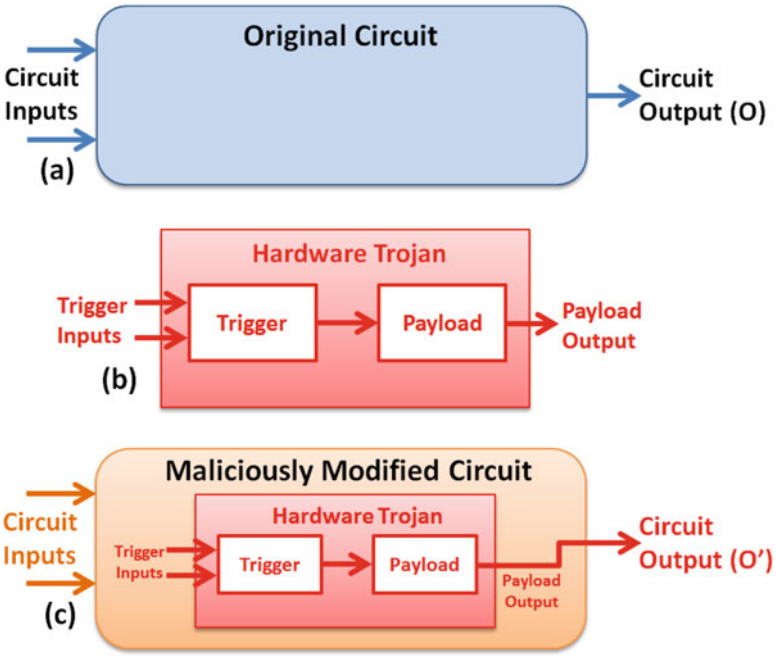

Fig. 2.1 Block diagram showing (**a**) original circuit, (**b**) simplified hardware Trojan, and (**c**) hardware Trojan inserted into the circuit, which inverts O to O' upon activation

Figure 2.1a shows a simplified block diagram of an original circuit, Fig. 2.1b shows the hardware Trojan with trigger and payload, and Fig. 2.1c shows the Trojan inserted into the circuit. When the Trojan activates in this example, its payload delivers a malfunction in the form of inverting the output of the circuit (i.e., changing the O to O').

Historically, hardware has been considered the *root of trust*, and the software or firmware that runs on top of the hardware has been untrusted until proven otherwise [6, 37]. However, much research has been performed over the last decade, and security researchers are aware of numerous Trojan models, attacks, countermeasures, as well as the threats to security and trust of the underlying hardware. Every entity (e.g., person, design house, foundry, supply chain) that is involved with the design, fabrication, testing, packaging, and delivery of an integrated circuit (IC) could be considered a potential adversary as they have the opportunity to tamper with the IC at multiple points in the design cycle. Therefore, the need exists to design "for security" in addition to specified functionality of the hardware.

A hardware Trojan is not a design or manufacturing fault. A fault (e.g., SA0, SA1, path delay) is an unintentional error or failure during these processes, and its location for activation is usually known by the designer. A Trojan, in any form, is an intentional insertion by an adversary, and its location for activation is unknown to

the designer. Although the intentions of a hardware Trojan are similar to its nefarious counterpart, the software Trojan, the hardware Trojan cannot be removed once the IC is fabricated, whereas the software Trojan can be eradicated post-deployment [6].

Trojans are activated by a specific mechanism, called a trigger, and deliver a specific function, called a payload. They can be small or large in size with respect to the rest of the circuit, ranging from just a few transistors to thousands of gates in a multimillion transistor SoC design [6]. Trojans exist in many forms and are most commonly triggered by a sequential or combinational digital circuitry (or a hybrid combination of both), but can also be triggered by analog stimuli. The payload can be digital or analog with each being specifically crafted to deliver malicious consequences upon activation.

Trojans are often undetectable to conventional pre-silicon and post-silicon manufacturing testing processes such as informal and formal verification [47]. This is due to test coverage for the hardware solely focusing on the specific functionality of the circuit, as well as exhaustive testing of all possible functions being both expensive and time-consuming. Trojans are intended to be stealthy and are inserted into rare internal nodes of the circuit, which reduces the likelihood of activation during normal testing. These nodes are often outside the scope (i.e., outliers or corner cases) of the circuits' intended functionality and are not activated during conventional test and verification methods. Unconventional methods do exist for detection, but these often require access to a verified and authentic (i.e., golden) IC or model for comparisons, and these methods may be performed as destructive (e.g., physical reverse engineering) or nondestructive (e.g., side-channel) testing methodologies [6, 37].

2.2 Trends, Tradeoffs, and Threats of Trojans

2.2.1 Semiconductor Design Flow

Over the last decade, there has been a shift in the global manufacturing model and design flow of companies that produce semiconductor ICs such as application-specific integrated circuits (ASIC), field-programmable gate arrays (FPGA), and system-on-chips (SoCs). These new trends are driven by various economic factors including monetary costs, time-to-market demands, and the increased complexity of semiconductors. Historically, hardware design companies performed the entire design flow as a trusted entity from "cradle-to-grave" including defining specifications, generating schematics and netlists, fabrication, testing, packaging, and delivery to the supply chain marketplace. This design flow was commonly referred to as a "vertical model." However, companies have been adopting a "horizontal model" in which they outsource certain steps of the design flow to untrusted entities and offshore foundries. Today, the majority of hardware design companies have fully adopted the horizontal model and now are rapidly moving toward a "fabless

model" in which they outsource all hardware fabrication. Many system integrators outsource both the hardware design and fabrication in order to maximize service to their customers and profits for their company.

The reliance on untrusted foundries reduces monetary costs for design companies as it eliminates the need to build and maintain a multibillion dollar fabrication facility. Using offshore foundries also allows design companies to have access to state-of-the-art fabrication technologies and reduce the risk of fabrication errors. By incorporating third-party intellectual property (3PIP) from untrusted entities into the hardware design, companies are able to decrease the time-to-market delivery and maximize the profit window of their product. These business decisions allow design companies to leverage the economies of scale created by the untrusted entities [39, 45].

However, these tradeoffs decrease the level of security and trust of the hardware, thus violating the traditional hardware *root of trust* philosophy. Relying on untrusted entities reduces control of the hardware design, thereby increasing the likelihood of a vulnerability being introduced during the design life cycle and supply chain distribution [45, 47]. These vulnerabilities exist in various forms including IP piracy, counterfeiting, cloning, overproduction, and hardware Trojans [6, 30, 37, 39, 40].

The modern horizontal model for semiconductor IC design and fabrication flow is shown in Fig. 2.2. Typically, a trusted entity (e.g., design house) is responsible for the specification, register-transfer level (RTL) design, netlist generation, and layout. An untrusted entity (e.g., foundry) is responsible for wafer fabrication, assembly, and testing. However, this generality is not always the case as demonstrated by the seven attack models described in Sect. 2.4.5.

2.2.2 Adversaries and Attacks

2.2.2.1 Adversarial Threats

Any entity that is involved with the design, fabrication, testing, packaging, or supply chain of an IC has the potential to be an adversary. Adversaries may be people, design houses, foundries, or even electronic design automation (EDA) or computer-aided design (CAD) software tools. What is common with all adversaries is the opportunity to tamper with the design (e.g., insert a Trojan) anytime the design is outside of the control of the rightful owner. These opportunities occur at the many stages of the IC's life cycle as shown in Figs. 2.2 and 2.3. The motivation of each adversary varies as does their design of the Trojan. Adversaries are motivated by many factors including monetary gains, increased market space in the supply chain, personal or political vendettas, tarnishing a competitor's reputation, or even the sole purpose of disrupting critical infrastructure through operational failures or leakage of sensitive information. Adversarial actions result in specially crafted malicious modifications to the original circuit, which are designed to achieve a specific nefarious goal.

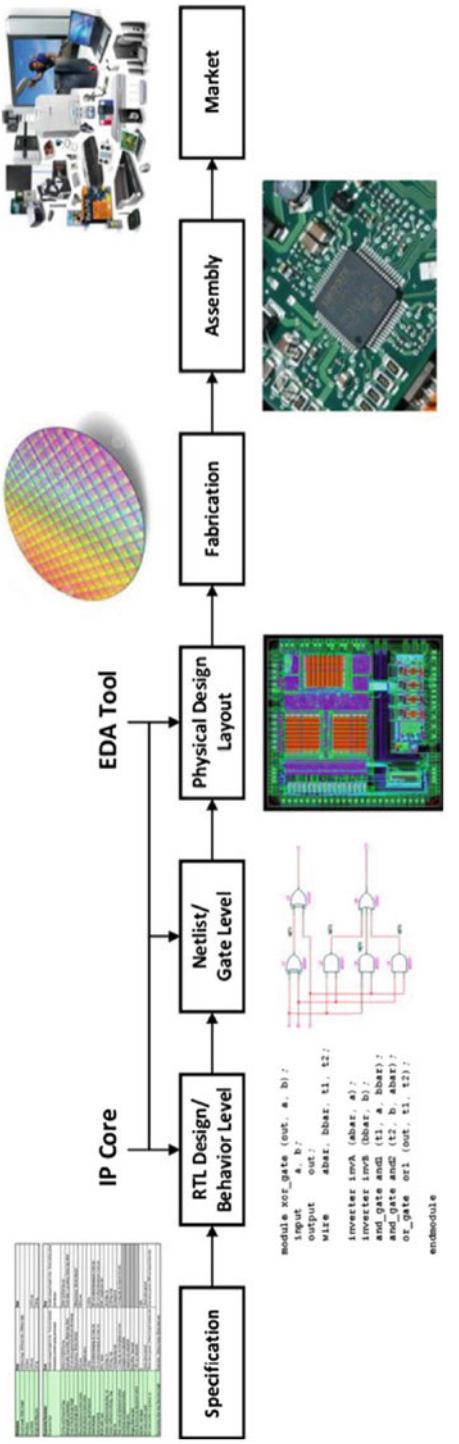

Fig. 2.2 Semiconductor IC design and fabrication flow [45]

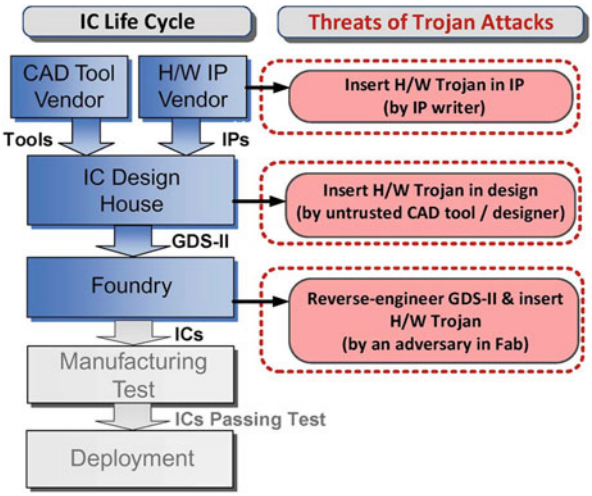

Fig. 2.3 Vulnerable stages in the integrated circuit (IC) life cycle [6]

2.2.2.2 Attack Surfaces

The goal of a hardware Trojan is as unique as its adversary. Trojans may be inserted into ICs including control circuitry, memory modules, sensors, and input/output drivers. They may also be inserted into embedded system processors allowing for software backdoors or inserted into cryptographic engines in SoCs to weaken, bypass, or disable the security features of the system [6]. Three examples of adversarial attacks that may occur during the design process are an untrusted third-party intellectual property (3PIP) vendor introducing a Trojan into the IP core of an SoC, an untrusted designer inserting a Trojan into unused cells of an IC, or even a trusted designer inadvertently inserting a hardware Trojan through the use of untrusted third-party EDA software tools.

Another example of attack surface is when an adversary at an untrusted foundry inserts a Trojan into the lithography mask during the fabrication process of the semiconductor wafer. This Trojan-infected wafer is then assembled into an IC and returned to the design company. Unless the design company has specific Trojan detection or prevention mechanisms for testing, such as a *golden IC* or *golden model*, the infected IC will enter the supply chain for integration into an unknowingly compromised electronic system. A *golden IC* or *golden model* is considered to be trustworthy as it has been verified to be fabricated exactly per the design specifications (i.e., nothing more, nothing less) and to be Trojan-free.

Furthermore, an adversary at a untrusted foundry or an untrusted design house could reverse engineer the entire IC design for the purpose of cloning (i.e., creating illegal copies) the IC. The adversary could insert a Trojan into the cloned IC and release the infected IC directly into the supply chain, thus bypassing any Trojan detection mechanism of the legitimate design owner. In this situation, there is no

golden model for verification, so integrators have to rely on other approaches such as self-referencing and side-channel analysis methods for Trojan detection [12, 23, 25].

Figure 2.3 shows examples of Trojan attacks during several vulnerable stages of the IC life cycle.

2.3 Comparisons and Misconceptions with Trojan Attacks

2.3.1 Trojans Compared with Bugs or Defects

Trojans should not be considered design bugs or manufacturing defects as this generalization is simply not accurate. Recall that *a hardware Trojan is an intentional and malicious modification of a circuit that is designed to alter the circuit's behavior in order to accomplish a specific objective*. A design bug is an unintentional problem (i.e., error) that is unknowingly introduced into the circuit during its design and development phases. A manufacturing defect is an unintentional physical phenomenon (e.g., imperfection) that occurs during the circuit's fabrication, assembly, and testing phases. Both design bugs and manufacturing defects can cause flaws, failures, or faults in the final assembled IC or electronic system.

Although design bugs and manufacturing defects lead to incorrect results or unexpected behaviors, these outcomes are detectable during conventional test and validation methods. That is, bugs and defects are discoverable with functional or structural testing as well as pre-silicon or post-silicon verifications. Similar to Trojan taxonomies and attack models (discussed in Sect. 2.4), companies also use models to detect bugs and defects. These models are typically bounded by the specification of the design and will support testing and verification only within the intended design of the circuit. For example, stuck-at-one (SA1), stuck-at-zero (SA0), opens, shorts, and path delay faults are all detectable based upon specification models, and their specific activation locations are usually identifiable within the circuit.

Trojans are similar to bugs and defect in that they all can lead to unwanted functionality. However, Trojans deliver an unwanted functionality that is not bounded by the specification of the design. As a result, Trojans are often undetectable with standard testing and validation practices. Since conventional models typically do not check for any functionality outside of the defined specification, adversaries often attempt to hide Trojans in internal circuit nodes that are difficult to reach, control, and observe during testing [38]. Trojans can also be designed to activate after a rare and arbitrary set of complex conditions have occurred [6, 37]. Further details of Trojan detection strategies are explained in Sect. 2.5.

Figure 2.4 illustrates the comparisons between hardware Trojans, design bugs, and manufacturing defects.

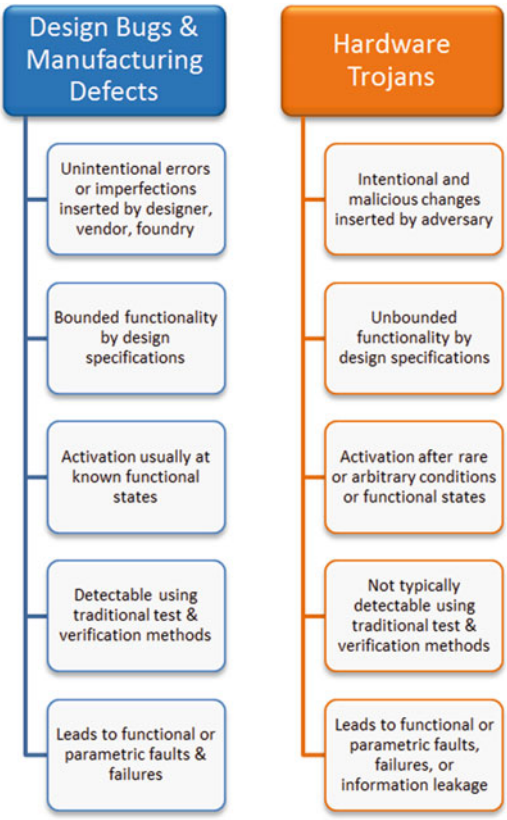

Fig. 2.4 Hardware Trojans compared with bugs and defects

2.3.2 Hardware Trojans Compared with Software Trojans

Trojans, whether software or hardware, share the same three key characteristics: malicious intention, evasion of detection, and rarity of activation. They also share a similar abstraction of two main components: a trigger and a payload. A software Trojan is commonly known as a computer program that contains specially crafted malicious code (i.e., payload) designed to cause harm to a targeted system once triggered. The malicious objective of software Trojans can include privilege escalation on the targeted system, leakage of sensitive user information including credentials and passwords, as well as data corruption, unauthorized encryption, and denial-of-service (DoS) attacks. Software Trojans are designed to remain stealthy, require specific events to occur for activation, and require special programs to detect and remove them. Most software Trojan detection programs include run-time monitoring, a concept that has been extended to hardware security (Sect. 2.5.1.2) [6].

Although the malicious intention of a hardware Trojan is similar to its nefarious counterpart, there are notable differences between these two Trojans. For example, software Trojans are hidden inside software code and are activated during program

Software Trojans	Hardware Trojans
Hidden inside software code and activates after specific conditions are met during program execution	Hidden inside physical hardware and activates after specific conditions are met during hardware operation
Spreads from user-to-user or adversary-to-user, through computer activities (e.g. file sharing or running infected programs)	Spreads from user-to-user or adversary-to-user, through physical insertion into circuits (e.g. infected ICs in supply-chain or untrusted entities)
Can be removed post-deployment (i.e. software updates)	Cannot be removed post-fabrication (i.e. no hardware updates)

Fig. 2.5 Hardware Trojans compared with software Trojans

execution, whereas hardware Trojans are hidden inside physical hardware and are activated after specific conditions occur during operation. Also, legitimate users of computer systems can unintentionally spread a software Trojan from user-to-user through routine activities, or adversaries can intentionally spread them to targeted or untargeted victims. This distribution can be accomplished in many ways such as through peer-to-peer file sharing or running infected programs from the Internet. Hardware Trojans, on the other hand, are typically spread from adversary-to-user since an IC is not easily replicated by the end-user. Another key difference is that hardware Trojans cannot be removed once the IC is fabricated, whereas the software Trojan can be removed post-deployment by way of local or remote updates to the program code [6].

Figure 2.5 provides a summary of comparisons between software and hardware Trojans.

2.3.3 Hardware Trojan Cause and Effect Misconceptions

The pressure on design companies to further reduce costs, decrease the time to market for product deliverables, and increase company profits has underpinned the semiconductor industry's shift toward horizontal and fabless business models. As a result, companies are relying more often on the acquisition and reuse of hardware third-party intellectual property (3PIP) as well as electronic design automation (EDA) and computer-aided design (CAD) software tools. This reliance is prevalent in the SoC industry where design specifications change many times during the design and manufacturing flow, and the hardware semiconductor companies must remain flexible to quickly respond to the market's demands. Unfortunately, these companies sometimes acquire 3PIP and design tools from untrusted entities (e.g.,

gray- or black-market vendors) without fully understanding the implications of their actions and the potential security risks to their SoC designs and to their customers.

Hardware Trojan attacks in hardware 3PIP are a serious security and trust risk that is difficult to mitigate. Companies and vendors that provide the trusted 3PIP (e.g., IP crypto cores) rarely make the golden model of their IP available to any outside entity. This results in a potential attack surface for adversaries: an untrusted entity (e.g., vendor, design house) can legally obtain a single version of the trusted 3PIP and insert a hardware Trojan into it, thus making it untrusted 3PIP. This infected 3PIP can be distributed to many naive SoC companies through various channels such as illegal file sharing services or other untrusted vendors. Since these SoC companies do not have the golden model that is required for conventional test and verification methods, they will have the virtually impossible challenge of verifying that the acquired 3PIP is secure and trustworthy [38, 47].

While it is true the SoC company can perform simulation of the 3PIP, this will only verify the functionality based on the design specifications; it will not guarantee the 3PIP is Trojan-free. The SoC company will not be able to compare their design, which is based on untrusted 3PIP, with the legitimate and trusted 3PIP. However, the work of [28] devised a technique to use the same hardware 3PIP acquired from several sources to reduce the risk and effects of a potential hardware Trojan. Also, the work of [47] developed a multiple-step methodology to identify and remove Trojans in 3PIP digital cores. Overall, the risk of using untrusted 3PIP still remains high as does its potential for delivering infected SoC products to customers and the supply chain.

Likewise, untrusted EDA tools present a similar risk of malicious modifications being inserted into the hardware design. Similarly to hardware 3PIP owners, EDA tool companies rarely make their golden model available to other companies. Untrusted EDA tools may be obtained through various methods such as illegal downloads, license key cracks, and untrusted vendors. It is a common practice for hardware design companies to use several tools from the same EDA tool suite, such as design automation, testing, and verification tools. Therefore, an adversary has several attack surfaces in which to maliciously modify the software tools to facilitate hardware Trojan insertion into a company's design.

Since hardware Trojans can be inserted into designs via the untrusted software EDA tools (e.g., through the hardware synthesis engine), detecting Trojans becomes more difficult as the design progresses. This results in the possibility of a hardware Trojan being inserted into a design early in the design flow via the untrusted EDA tool, which is later ignored or not detected by a different, albeit trusted, tool from the same vendor. The work of [26] uses the security paradigm of a completely specified design and low latency observability in order to design trustable hardware using untrusted software tools. However, again similar to untrusted 3PIP, using untrusted software EDA tools still runs the risk of delivering infected hardware products to customers and the supply chain [6].

Another popular misconception is that multiple untrusted entities result in improved security and trust. However, this allows for malicious collusion between multiple untrusted entities, also known as a multilevel attack, and results in a form

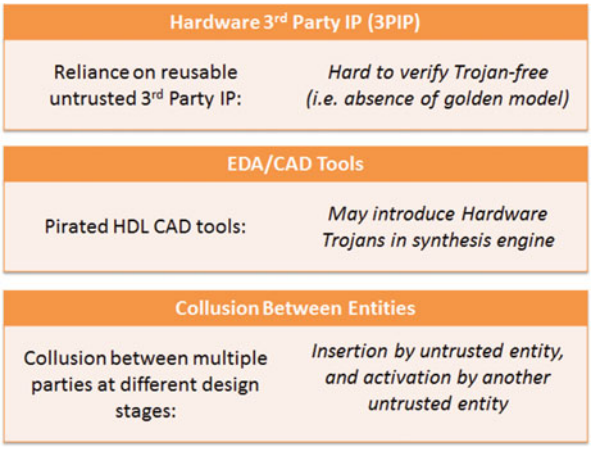

Fig. 2.6 Cause and effect misconceptions related to hardware Trojans

of risk that is difficult to defend against. Collusion can occur throughout different untrusted stages of the hardware design flow and life cycle [6]. For example, the work of [1] demonstrates a hardware Trojan inserted by an untrusted entity that is activated with a specific fault condition known only to the other colluding party. As another example, the work of [19] demonstrates a hardware Trojan designed to leak sensitive information through an analog side-channel. This Trojan is activated by another untrusted entity who also obtains and analyzes the leaked information. Collusion can also occur between multiple teams inside the same vendor. For example, the work of [27] illustrates this point and provides a codesign approach for preventing collusion between multiple rogue insiders at the design house. Although anti-collusion techniques do exist, the work of [1] demonstrates that multilevel collusion between untrusted entities results in a significantly stronger adversarial threat than what results from only a single adversary.

Figure 2.6 provides a summary of the cause and effect misconceptions discussed in this section.

2.4 Offensive Strategies

2.4.1 Taxonomy of Trojan Types

In order to properly model hardware Trojans for offensive and defensive postures, one must first understand the different types of Trojans. The categories of these types, referred to as taxonomies, represent the framework for classifying hardware Trojans based on their individual characteristics and also provide the foundation for building metrics to evaluate Trojan detection mechanisms. The first published hardware Trojan taxonomy was proposed in 2008 by [43] and consisted of six attributes including three principle categories based upon physical, activation, and

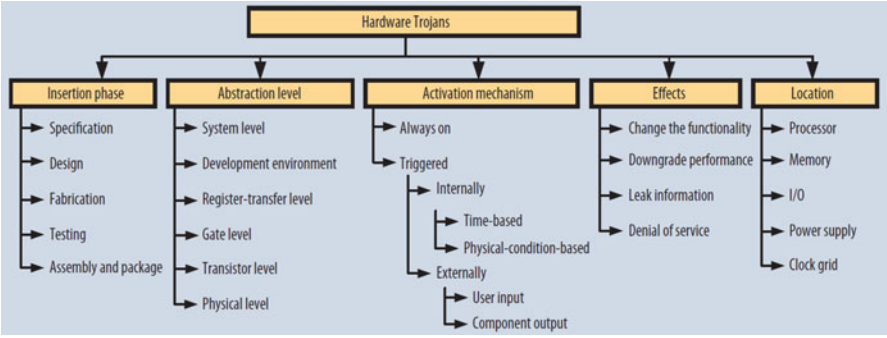

Fig. 2.7 Five comprehensive hardware Trojan taxonomy categories [17]

action characteristics. As hardware Trojans became more complex, this elemental taxonomy was improved to comprise of nine attributes for the same three principle categories [36]. The most comprehensive taxonomy to date is the work of [17] that comprises of five categories (insertion phase, abstraction level, activation mechanism, effects, and location), with each category containing multiple attributes. This taxonomy is predicated on two key criteria: (1) coverage (it should classify any-and-all Trojans) and (2) resolution (it should separate significantly different capabilities of Trojans). This comprehensive taxonomy is shown in Fig. 2.7.

The *insertion phase* describes the stages of the design and fabrication life cycle where the hardware is vulnerable to malicious modification. This phase ranges from defining the hardware characteristics (i.e., design specifications) to physical IC placement (i.e., assembly) on a printed circuit board (PCB).

The *abstraction level* describes the various development stages of the hardware IP prior to fabrication. This level spans from the physical dimensions and locations of the internal components in the circuit (i.e., physical level) to the final definitions of the interconnects and communication protocols used in the IC (i.e., system level).

The *activation mechanism* describes the means by which the Trojan is triggered. This includes always-on Trojans such as those continually leaking information through EM radiation, as well as Trojans requiring specific triggers for activation such as internal sequential counters or external triggers from input data streams.

The *effects* category describes the unwanted result from the Trojan's delivered payload. This ranges from introducing small errors that are difficult to detect (i.e., change the functionality) to full consumption or failure of hardware resources, thus preventing system availability (i.e., denial of service).

The *location* category describes where inside the hardware a Trojan can physically be inserted. This category spans from a single Trojan targeting a single component (e.g., system clock) to perform fault-injection attacks to multiple distributed Trojans targeting multiple complex components (e.g., processors) to alter the order of instruction execution.

This comprehensive Trojan taxonomy has been validated against a total of 56 hardware Trojans for the coverage and resolution criteria described in this section,

and it correctly captured all of the Trojans into the proper categories [17]. However, as with many aspects of security and trust, continuous advances are required to be made in order to stay ahead of the adversary. The website https://www.trust-hub.org maintains a collection of hardware Trojan benchmarks that have been developed and updated by researchers in the hardware security and trust community [32, 34, 41].

From 2007 through the present day, there has been much research focused on hardware Trojan modeling, circuit generation, and benchmarks [32, 34, 36, 41, 45]. However, within just the last few years, the number of Trojan design publications has started to trend downward, possibly indicating that Trojan designs have saturated. On the other hand, the number of research publications focused on countermeasures, such as detection and prevention, has greatly increased. These trend changes may be due to the fact that there exist a virtually unlimited number of different Trojan designs, and the critical research needs to be focused on the defensive postures with prevention mechanisms possibly outweighing detection mechanisms [45].

2.4.2 *Taxonomy of Trojan Triggers and Payloads*

The fundamental taxonomy of triggers and payloads, which are the two main components for hardware Trojans, is shown in Fig. 2.8. The trigger continually monitors specific signals in the circuit and activates when an expected event occurs, which is typically derived from the original circuit. The payload is activated by the trigger and delivers the malicious behavior to the circuit. Trojans may remain undetected and inactivated for many years while waiting for the specific circumstance to occur before they trigger and deliver their nefarious payload to the circuit.

Trojan triggers occur in two types: digital or analog. Digital is the most commonly researched Trojan as it consists of combinational and sequential circuits.

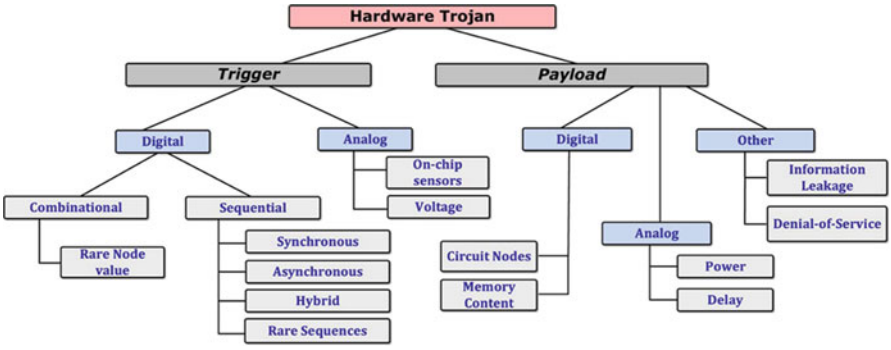

Fig. 2.8 Taxonomy of hardware Trojan triggers and payloads [6]

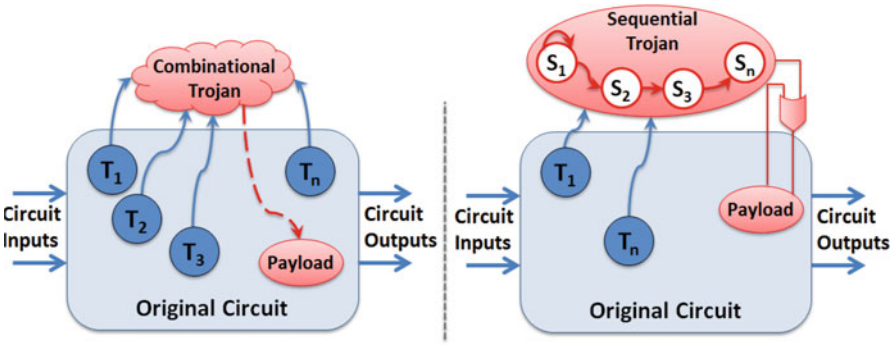

Fig. 2.9 Generic models of combinational and sequential Trojans

Combinational Trojans are stateless, meaning they contain no state elements (e.g., flip-flops, latches), and they rely on a specific condition occurring at a specific set of rare nodes within the circuit. Sequential Trojans are stateful, meaning they rely on a specific sequence of states to be traversed before activation (e.g., counters, finite state machines). Sequential Trojans are more difficult to detect since they require a number of arbitrary conditions to be met before activation, which can become computationally infeasible to detect with conventional testing methods. Analog triggers, on the other hand, rely on various natural phenomena (e.g., temperature, RF radiation, gate capacitance) for activation. A hybrid Trojan is a combination of a digital and an analog Trojan. Two generic Trojan models can be seen in Fig. 2.9.

Recall that Trojans should be virtually undetectable and rarely activated. Therefore, an adversary would choose nodes that are unlikely to activate during conventional testing methods. Upon activation, the Trojan's payload will be delivered to the circuit. The payload can be categorized as digital (e.g., affecting logic values, opening backdoors) or analog (e.g., affecting performance, EM emissions) or others (e.g., acceleration of IC aging, leaking information). The payload is the critical part of the Trojan as it is ultimately what modifies the original behavior of the circuit. Examples of fundamental hardware Trojans with triggers and payloads are shown in Sect. 2.4.3.

2.4.3 Fundamental Trojan Examples

Figure 2.10 shows a combinational Trojan with a NOR gate as a trigger and an XOR gate as the payload. This Trojan is activated only when the specific condition of $A = 0$ and $B = 0$ occurs at the NOR gate's trigger nodes, resulting in the payload delivering an inverted output *Cmodified*.

Figure 2.11 shows a synchronous sequential Trojan (a.k.a. Trojan "time bomb") with a simple counter for activation. The trigger consists of a $k - bit$ counter and an

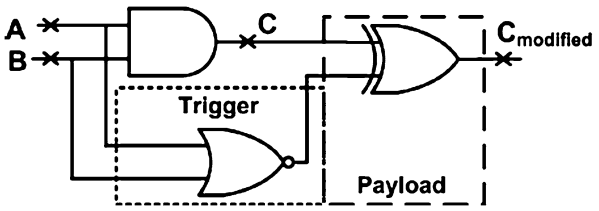

Fig. 2.10 Combinational Trojan circuit [9]

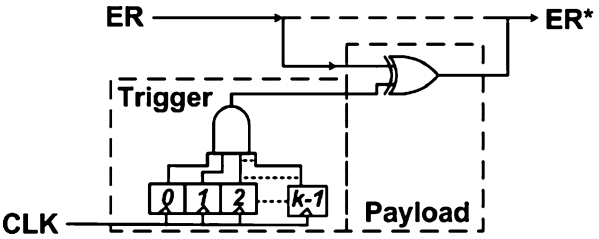

Fig. 2.11 Sequential Trojan circuit [9]

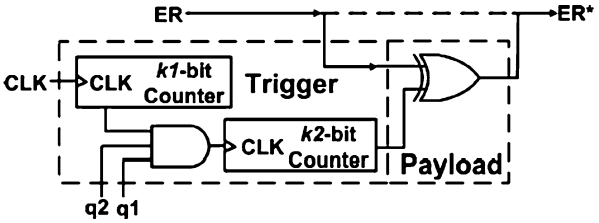

Fig. 2.12 Hybrid Trojan circuit [9]

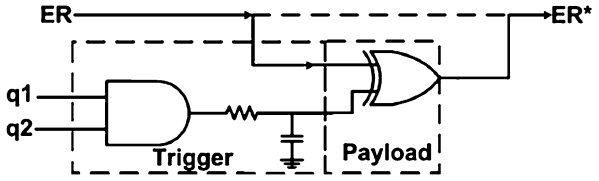

Fig. 2.13 Analog Trojan circuit [9]

AND gate, and the payload consists of an XOR gate. This Trojan is triggered after a predefined $2^k - 1$ counts, resulting in an inverted output *ER**. An asynchronous version of this Trojan can be created by substituting the clock (*CLK*) with another logic implementation.

Figure 2.12 shows a hybrid Trojan consisting of both synchronous ($k1$-bit) and asynchronous ($k2$-bit) counters. Both of these counters must reach their predetermined values for activation, resulting in an inverted output *ER**.

Figure 2.13 shows an analog Trojan in which a capacitor is charged if the AND gate output (driven by $q1$ and $q2$) has been set to 1 for a calculated period of time, resulting in an inverted output *ER**. If the AND gate output is not set high for the correct duration of time, the capacitor will discharge to ground and the Trojan will not activate.

The previous examples have shown hardware Trojans with digital or analog triggers delivering a digital payload. Recall that digital payloads are designed to

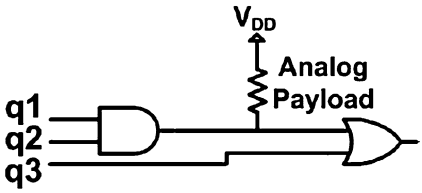

Fig. 2.14 Trojan with analog payload to Vdd [9]

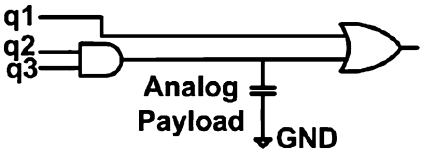

Fig. 2.15 Trojan with analog payload to GND [9]

affect targeted logic values at specific internal nodes, whereas analog payloads affect characteristics such as performance. Figures 2.14 and 2.15 offer examples of analog payloads.

Figure 2.14 shows an analog payload in which a fault is created, via the resistor to Vdd, when the output of the AND gate is logic zero.

Figure 2.15 shows an analog payload in which path delay is affected, via the capacitor to GND, when the output of the AND gate is logic one.

2.4.4 Innovative and New Trojan Attacks: Designs and Examples

Hardware Trojans have evolved to become more potent threats against secure and trustworthy hardware. Trojan attacks now include more advanced functions such as temperature-driven activation, radio, power, and optical side-channel leakage of information, accelerated aging of ICs, as well as denial-of-service (DoS) attacks against device availability [15, 18, 43]. This section explores several examples of innovative and recent designs of hardware Trojan attacks.

2.4.4.1 Side-Channel Trojan

Malicious off-chip leakage enabled by side-channels (MOLES) [19] demonstrates a hardware Trojan that is inserted by an untrusted foundry. MOLES leaks sensitive information outside of the IC through an analog side-channel, which is later acquired and analyzed by another untrusted entity. MOLES was designed to compromise hardware security modules (HSMs), which are the tamper-resistant cryptographic engines for embedded systems and general-purpose computers. It was implemented in an Advanced Encryption Standard (AES) IP core to leak

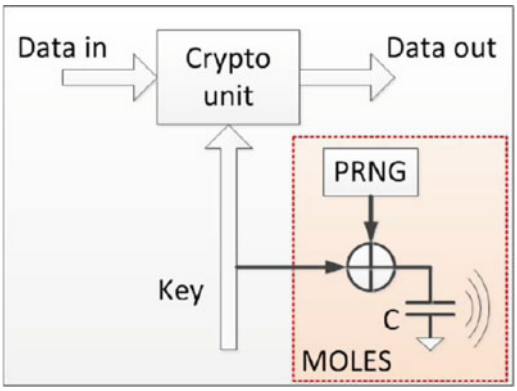

Fig. 2.16 MOLES circuit embedded inside a hardware security module [19]

multi-bit cryptographic keys. All of the bits were leaked with a signal-to-noise ratio (SNR) below the noise power level of the infected IC in order to remain hidden while the Trojan is activated. Also, MOLES was designed to be very small in size (i.e., fewer than 50 gates) in order to evade detection from automatic test pattern generation (ATPG) testing and layout inspection processes. Figure 2.16 shows the block diagram of MOLES.

2.4.4.2 Semiconductor Trojan

Stealthy dopant-level hardware Trojans are described in the work of [4] in which the dopant polarity was altered in the existing transistors of the infected IC. This type of Trojan is inserted by an untrusted foundry after placement and routing occurs during the layout level. Since there is no additional circuitry required for this Trojan, the appearance and functionality of the IC are not changed. Therefore, the malicious modification is virtually undetectable to optical inspection techniques and can defeat golden IC model verification methods. This Trojan was inserted into a case study model of a cryptographically secure processor in order to reduce the entropy of the random number generator (RNG) used to produce the cryptographic keys. The authors claim their Trojan will allow the compromised processor to still pass the built-in self-test (BIST) as well as the National Institute of Standards and Technology (NIST) test suite that is commonly used to rate the quality of random number generation. Figure 2.17a shows an unmodified inverter gate, and Fig. 2.17b shows a dopant Trojan inserted into the inverter gate resulting in a constant output of V_{DD}. A close inspection of this figure reveals the contact, metal, and polysilicon areas are indeed identical in both cases. The only change is to the polarities of the *N* and *P* dopants.

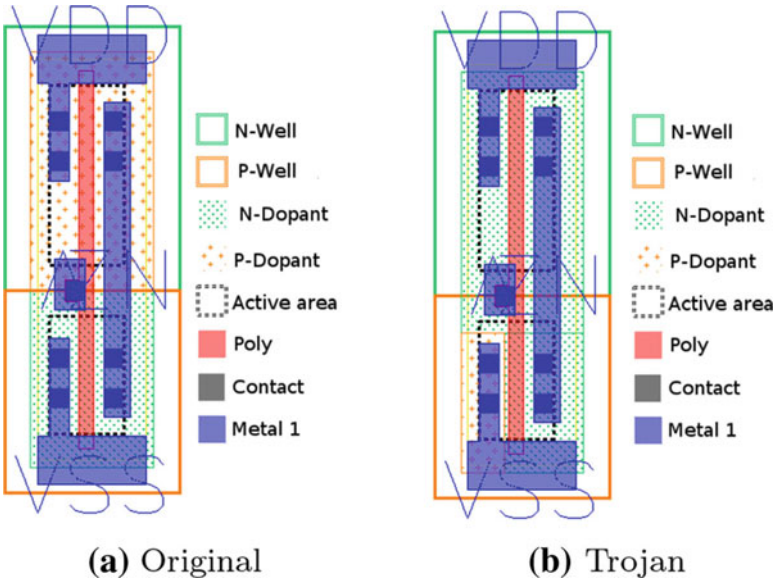

Fig. 2.17 Layout of (**a**) Trojan-free inverter gate and (**b**) inverter gate with Trojan to output constant V_{DD} [4]

2.4.4.3 Analog Trojan

Analog malicious hardware, called "A2", demonstrates how an untrusted foundry can insert an analog hardware Trojan into empty cells of a circuit [16]. A2 is a capacitor-based Trojan that slowly diverts charge from internal connections that rarely toggle their digital values. Once A2's capacitors reach a specified charge state, the Trojan triggers and delivers a payload that overrides a flip-flop's current state, thus forcing it to a predetermined value. Although A2 is a hardware Trojan, its objective is to enable a remotely controlled software privilege escalation attack by forcing a targeted bit in a security register to a specific value. A2 was implemented into an open-source CPU processor just prior to the CPU being fabricated, and the Trojan attacks were successful. Essentially, A2 implements an analog counter as a trigger, meaning that it does not require numerous additional gates as does a conventional digital counter-based trigger. A2 can be as small as one gate and is more stealthy than its digital counterpart; thus it is more elusive to functional verification, simulation, and side-channel Trojan detection methods. Figure 2.18 shows the behavior of the A2 Trojan, which takes multiple rising-edge triggers to charge the capacitor to the threshold voltage value required for activation.

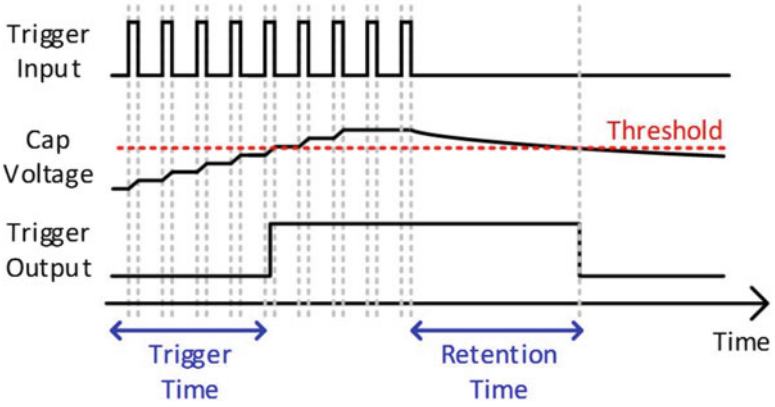

Fig. 2.18 "A2" analog Trojan circuit behavior. Notice capacitor requires multiple triggers prior to activation [16]

2.4.4.4 Digital Trojan

Hardware Trojans can also exist in digital finite state machines (FSMs) as demonstrated in [13]. Vulnerabilities exist in high-level incomplete design specifications, which can be exploited by adversaries during the design flow, as well as in post-fabrication through inadvertent trapdoors found in the defined FSM. When the number of protected states is not a power of 2 (as in n^2 states), there will be unused states in the FSM that will be treated as "*don't-care*" conditions. The adversary inserts the logic Trojan into the *don't-care* conditions inside a sequential FSM (i.e., where the next state or output is not specified), which allows an attacker access to protected states in the FSM once the Trojan is triggered. If a *don't-care* condition is not Trojanized, logic design tools may use it for optimization. However, at the circuit level, *don't-care* conditions will be assigned a deterministic next-state value by the EDA/CAD tools which may have been coerced by the Trojan. Since the Trojan is inserted early in the flow, it may pass undiscovered by Trojan detection mechanisms used later in the flow. Figure 2.19a shows a four-state transition graph, which was originally a three-state FSM with state "11" as a *don't-care*. The solid blue edge lines represent the defined state transitions, and the dashed orange lines represent the *don't-cares* in the FSM. The dashed orange edge lines will be implemented by the circuit (Fig. 2.19b) with their assignments allowing an attacker potential access to protected states in the FSM.

2.4.4.5 Other Notable Trojans

2.4.4.5a: Insertion of hardware Trojans into the silicon of a fabricated wireless cryptographic IC was accomplished in the work of [20]. These Trojans were inserted

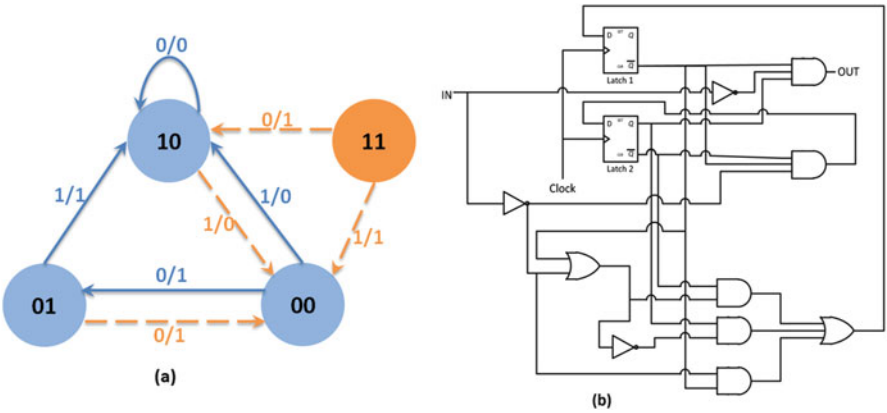

Fig. 2.19 Vulnerable four-state FSM (originally three states shown in blue) showing (**a**) dashed orange edge lines and state "11" representing *don't-care* conditions as a result of the digital logic implementation (**b**) where the *don't-cares* may allow an attacker access to protected FSM states [13]

into the AES core and the ultra-wideband (UWB) transmitter of the application-specific integrated circuit (ASIC) chipset. This attack resulted in encryption key leakage that was concealed within the amplitude and frequency design margins allowed due to fabrication process variations.

2.4.4.5b: A software-exploitable hardware Trojan was implemented inside an embedded processor as demonstrated by [44]. This hardware Trojan was modeled by a sequential FSM and was triggered with a specific combination of firmware instructions and data processing sequences. The attack resulted in leakage of the program IP, the encryption key, and also caused the system to malfunction.

2.4.4.5c: Reliability Trojans, which are malicious alterations of manufacturing conditions during semiconductor fabrication, are introduced in [35]. These Trojans exploit the wearing-out mechanisms for CMOS transistors, such as negative bias temperature instability (NBTI) and hot carrier injection (HCI). The attack objectives include reduced reliability, accelerated aging, and premature failure of ICs such as SRAM cache memory, all of which are triggered only with time and usage of the IC.

2.4.4.5d: Small, optimized, and performance-based Trojans have been designed to evade detection in the works of [7, 42]. These low-impact Trojans rely on modifications such as resizing of logic gates, interconnect tampering, and insertion of resistive bridging faults at single failure points in a circuit. They are designed to be inserted without impacting path delays, power consumption, or area overhead of the circuit. Their attack objectives include privilege escalation, erratic behavior, incorrect outputs, and hardware failure.

2.4.4.5e: Field programmable gate arrays (FPGAs) and systems on chip (SoC) are also vulnerable to hardware Trojan attacks. The work of [33] developed a com-

prehensive taxonomy of Trojan attacks in FPGAs. The work of [11] demonstrates an automated security analysis framework designed to detect hardware exploits including Trojan attacks in SoCs. Also, the work of [34] demonstrates an extensive database of hardware Trojan attacks which can be used to be insert Trojans into FPGAs and SoCs for research of defensive security and trust techniques.

2.4.5 *Trojan Attack Models*

The proper modeling of hardware Trojans is essential to accurately categorizing the adversarial threats and analyzing the effects of Trojan attacks. Prior to developing an attack or countermeasure method, the adversary or defender must consider the appropriate Trojan model. Recall that adversaries can insert Trojans into hardware during many phases of the design flow, which leads to the need for multiple attack models. Accurate Trojan attack models must be based on the entire semiconductor supply chain. For SoCs, this can be divided into three phases: IP core development, SoC development, and fabrication. These three phases result in three types of entities that can potentially attack the hardware design: 3PIP vendors, SoC developers, and fabrication foundries. This adversarial threat modeling concept can be extended to other ICs, and existing research has been performed to categorize different models of Trojans attacks [17, 29, 36]. The work of [45] provides us with seven comprehensive attack models for SoCs, shown in Fig. 2.20.

A summary of the seven comprehensive Trojan attack models follows:

Model A: Untrusted 3PIP Vendor—Most SoC designers have to acquire some forms of third-party IP (3PIP) cores to complete their designs. This is driven by demands for reduced costs, faster time to market, and decreases in physical IC size coupled with increased functional complexity. Adversaries at the untrusted vendor can insert hardware Trojans into the 3PIP without the SoC designer's knowledge.

Model	Description	3PIP Vendor	SoC Developer	Foundry
A	Untrusted 3PIP vendor	Untrusted	Trusted	Trusted
B	Untrusted foundry	Trusted	Trusted	Untrusted
C	Untrusted EDA tool, or rogue employee	Trusted	Untrusted	Trusted
D	Commercial off-the-shelf (COTS component	Untrusted	Untrusted	Untrusted
E	Untrusted design house	Untrusted	Untrusted	Trusted
F	Fabless SoC design house	Untrusted	Trusted	Untrusted
G	Untrusted SoC developer with trusted IPs	Trusted	Untrusted	Untrusted

Fig. 2.20 Seven comprehensive hardware Trojan attack models [45]

Model B: Untrusted Foundry—Most design houses are partially or fully fabless, meaning they outsource the fabrication of their ICs to offshore and untrusted entities. The outsourcing decisions are a tradeoff between the need for the latest fabrication technologies at the lowest costs and the security of their designs. Adversaries at these untrusted foundries have access to all layers of the design and are able to insert Trojans into any of the lithography masks, as well as reverse engineer the design for counterfeiting purposes.

Model C: Untrusted SoC Developer—Highly trained SoC designers and specialized design tools are required to produce complex hardware designs. Adversaries in this model are insider threats, and they may use untrusted (i.e., pirated) CAD and EDA software tools.

Model D: Untrusted COTS Components—Many commercial off-the-shelf (COTS) components are used in designs. COTS items are less expensive than a custom product, and they typically do not require custom development for integration into a system. These items are developed in a completely untrusted manner resulting in multiple vulnerable stages in the design flow.

Model E: Untrusted Design House—Designs in this model are fabricated in a trusted foundry, but the design house and 3PIP vendors are not trusted to produce Trojan-free designs. Outside of the trusted foundry, the entire supply chain is untrusted.

Model F: Untrusted Outsourcer—This is a combination of Model A and Model B, and it applies to almost all fabless IC design houses. These designers use 3PIP vendors and untrusted foundries resulting in the inability to guarantee a Trojan-free hardware design.

Model G: Untrusted Systems Integrator—This model includes untrusted system integrators catering to a variety of customers who wish to have a supplier capable of both design and fabrication. This developer can pull from a variety of resources to meet customer demands; however vulnerabilities may be introduced into the completed hardware design.

The work of [45] also included a comprehensive analysis of 161 published papers on countermeasures against these types of Trojan attack models. The results show a remarkable 89% of the countermeasure papers covered Model F (untrusted outsourcer), and almost 60% of the published papers covered Model B (untrusted foundry). Model A (untrusted 3PIP vendor) was covered in almost 30%, and Model C (untrusted SoC developer) was covered in 13% of these papers. It is worth noting that Models D, E, and G accounted for virtually 0% of the papers as these three models can be absorbed by other attack models.

It is widely known that Trojan attacks typically occur in untrusted entities. Consequently, countermeasures should be performed only by trusted entities. These seven attack models show that foundries, vendors, and designers all play the role of either untrusted or trusted entities, but they cannot be both roles at the same time.

2.5 Defensive Strategies

2.5.1 Taxonomy of Trojan Countermeasures

It is very difficult to detect hardware Trojans using conventional test and validation processes, as introduced in Sect. 2.3.1 and analyzed in Sect. 2.5.5. Whether performing pre-silicon or post-silicon verification or executing structural, functional, or random test patterns, these conventional approaches perform poorly for detecting hardware Trojans. These processes are designed to detect defects in manufacturing workmanship and typically only test for expected operating conditions within the circuit. Trojans, by nature, are hidden among rare internal nodes that are not normally activated during device testing. An adversary can choose from an extremely large selection of Trojans to insert into the circuit, whereas performing deterministic and exhaustive defensive testing for all conditions is computationally infeasible. Additionally, parametric parameters such as path delay, internal noise, and power consumption are different for each IC based upon manufacturing tolerances, thus making detection inherently more challenging [3, 6, 9, 36, 43, 45, 48]. Therefore, the need exists for both detection and prevention mechanisms to defend against hardware Trojans.

Hardware security and trust researchers have developed three broad categories of countermeasures for hardware Trojans and have proposed the taxonomy shown in Fig. 2.21. The three main categories of countermeasures are Trojan detection, design for trust, and split manufacturing for trust [45]. The single letters in the blocks of this taxonomy cross-reference the listed countermeasure in this taxonomy with its particular attack models that were discussed in Sect. 2.4.5.

2.5.1.1 Trojan Detection

The goal of Trojan detection is to verify hardware designs without requiring supplemental circuitry [45]. Additional circuitry would lead to an increase in manufacturing cost, circuit size and performance, and power overhead. Trojan detection is performed during pre-silicon and post-silicon design stages. *Pre-silicon verification* helps SoC designers to validate 3PIP in the final design stages prior to fabrication. This includes performing functional validation, structural and code analysis, and formal verification. Although these techniques are good for identifying unintentional design errors and manufacturing faults, they offer no guarantees against hardware Trojans. *Post-silicon verification* provides more guarantees against Trojans, but at a different cost, which can be further categorized into destructive and nondestructive methods [9, 45].

Destructive methods (e.g., depackaging of ICs, physical reverse engineering) offer the highest assurance against Trojans as the fabricated IC can be visually verified against a golden IC or golden model. It is a complicated process involving

layer-by-layer de-metallization using a chemical mechanical polishing (CMP) technique followed by image reconstruction and analysis using a scanning electron microscope (SEM). This method allows for the identification of individual gates, transistors, and routing elements contained within the IC. This process is performed on a one-by-one basis taking several weeks to conduct, and the IC is destroyed during this process. Additionally, an adversary can insert a Trojan into only a small number of ICs instead of the entire lot, thus making destructive detection methods impractical from a scalability perspective. However, destructively testing a limited number of ICs is still beneficial as the information gained from the samples may be used to form golden models for verification with other Trojan detection methods (e.g., side-channel analysis) [6, 9, 45].

Nondestructive methods include functional testing and side-channel analysis to identify possible hardware Trojans. *Functional testing*, also known as *logic testing*, is a method of inputting specific logic patterns into the IC with the goal of triggering any existing Trojans. This form of testing is not equivalent to conventional testing as those test vectors aim to identify bugs and defects. Trojans are typically hidden in low-controllable and low-observable nodes, thus making them difficult to reach with conventional testing methods. *Side-channel analysis* is a method of acquiring and analyzing characteristics that are unique to each IC. The goal is to identify additional circuitry containing Trojans through observing changes in the physical parameters of each IC. This method includes performing analysis on consumed and leaked power, gate timing, path delay, circuit temperature, and electromagnetic radiation. Side-channel analysis may use ring oscillators, shadow registers, and delay elements to detect fluctuations indicative of hardware Trojans. Nondestructive methods can be performed numerous times and on as many ICs as desired. Although functional testing and side-channel analysis typically require a golden IC or model, together they form a complementary approach toward nondestructive hardware Trojan detection [6, 9, 36, 43, 45].

Functional validation is a form of pre-silicon Trojan detection with the main idea being similar to functional testing (i.e., logic testing). Functional validation is performed using modeling and simulation, which requires no physical connections to the device under test (DUT). Functional testing, on the other hand, requires physical connection to the DUT and is usually performed on a specialized test stand. The test stand is required to apply all of the generated test input vectors (based on the design specifications) and to collect the output of the device. Functional testing methodologies can be applied to functional verification, but not vice versa [45, 47].

Formal verification is a pre-silicon or pre-synthesis design verification technique, which is typically performed to confirm that a circuit has been indeed designed in the precise manner as defined by the design requirements. In the context of security and trust, formal verification is a mathematical approach toward exhaustively validating the entire security and trust specification of an IC. It includes using a specified set of security policies and proof-checking methods. Formal verification for Trojan detection is based on three verification methods: property checking, equivalence checking, and model checking. Respectively, these methods allow for verification of requirements in hardware test bench properties, checking equivalence between

RTL, netlist, and GDSII files, as well as checking the models used for system-level languages (e.g., Verilog and VHDL) against the defined security specifications [45, 47].

Code and circuit coverage is another pre-silicon Trojan detection technique. Analysis of hardware description language (HDL) may be performed structurally or behaviorally to identify rare and redundant internal nodes within the HDL code and the circuit. The coverage typically includes RTL line execution, FSM reachability coverage, and coverage of gate-level netlists combined with functional assertions indicating success or failure. This analysis aims to locate and identify the nodes with the highest probability indicative of a location for Trojan insertion. Quantitative metrics and manual post-processing of the results may be used to identify unusually rare nodes or gates with low observability, low reachability, and low probability, all of which are ideal targets for adversaries to insert hardware Trojans [45].

2.5.1.2 Design for Trust (DfT)

An alternative to the Trojan detection methods described above is to design for trust (DfT). DfT is a method that integrates security and trust throughout the entire design and manufacturing flow. It includes facilitating detection, preventing Trojan insertion, and trustworthy computing on untrusted components as shown in Fig. 2.21.

The first approach to DfT is to *facilitate detection* by incorporating functional testing, side-channel analysis (both previously described), and run-time monitoring.

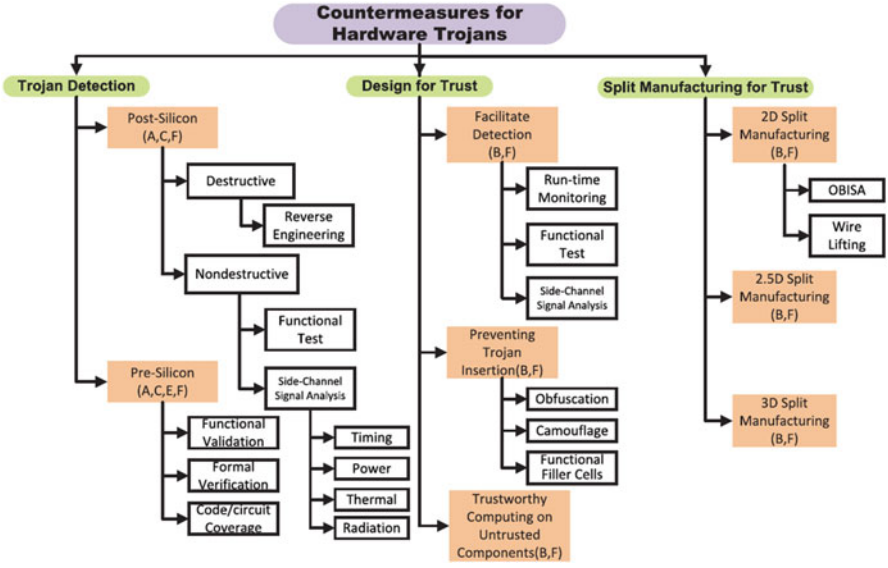

Fig. 2.21 Taxonomy of Trojan countermeasures [45]

Run-time monitoring can increase the trustworthiness of hardware by continually monitoring all critical computations for abnormalities, thus reducing the effect of Trojan attacks. Run-time monitoring can detect malicious behavior and automatically disable or bypass the malicious logic allowing the IC to restore reliable operation. It uses existing on-chip and supplemental off-chip (e.g., online) resources to continually monitor characteristics of the IC including behavior, operating conditions, transient power, and temperature [6, 9, 24, 36, 45].

Preventing Trojan insertion is another approach to DfT and includes obfuscation, camouflaging, and functional filler cells. *Obfuscation* is a method of hiding circuit functionality by inserting additional logic-locking circuitry into the design in order to conceal the correct functionality and the intended hardware design. The obfuscated circuit regains full functionality when the correct logic key is applied to the inputs. Obfuscation is effective for Trojan prevention in combinational, sequential, and reconfigurable logic. *Camouflaging* is a method of creating indistinguishable layouts of gates by using additional dummy contacts and fake interconnects between layers of the different gates in the circuit. Camouflaging prevents the attacker's ability to reverse engineer the circuit's netlist, thus preventing Trojan insertion. *Functional filler cells* are a method of inserting functioning gates into empty spaces in the hardware design. Typically, EDA/CAD tools fill empty spaces with non-functioning standard cells, thus allowing an attacker to replace these unused cells with a Trojan. Functional filler cells make use of all empty locations by inserting functional gates to form specified combinational logic, which can be tested during the design flow. A failure in the functionality of the filler cells can be indicative of an inserted Trojan [6, 45].

Another method for DfT is by performing *trustworthy computing on untrusted components*. This method is inherently resilient to Trojan attacks, which separates it from other prevention methods such as run-time monitoring and obfuscation. This method mitigates the effects of activated Trojans as the trusted computing software, which is running on the untrusted hardware, can be distributed over multiple independent mechanisms and processes. These risk reduction methods include distributed software scheduling over multiple multicore processors, using the identical untrusted 3PIP from different untrusted vendors, and comparing multiple 3PIP sources with similar untrusted designs [45].

An important point for hardware designers to consider is the balance between hardware Trojan detection and prevention methods with the actual need for increased security and trust of the hardware design. For Trojan detection, as the size of a circuit increases, so does the number of additional gates and internal nodes. These additional gates may unintentionally introduce low-controllable and low-observable nodes, which makes Trojan detection even more challenging. In addition, intentional modifications of gates for Trojan prevention may negatively impact the performance of the circuit. These impacts occur in many forms including path delays, power consumption, and area overhead of the circuit. As with design specifications, this inevitable tradeoff must be considered throughout the entire design flow and manufacturing process.

2.5.1.3 Split Manufacturing for Trust

Recently, split manufacturing has been offered to protect against hardware Trojans. This manufacturing process divides the hardware design into front-end-of-line (FEOL) and back-end-of-line (BEOL) sections that are fabricated by different foundries. Typically, an untrusted foundry will fabricate only the FEOL portion of the design and then ship their wafer sections to a trusted foundry who fabricates the BEOL section and integrates both sections. The process is conducted in this fashion as FEOL fabrication is of higher cost (i.e., monies, machinery, time) than BEOL fabrication. Split manufacturing prevents the untrusted foundry from having access to all layers of the IC, since having this information would allow an adversary to easily insert Trojans.

Common techniques of split manufacturing include *2D integration* (described above), 2.5D integration, and 3D integration as shown in Fig. 2.21. In *2.5D integration*, the design is divided into two sections (FEOL and BEOL), which are both fabricated by the untrusted foundry. A middle section, called a silicon interposer as it contains the interchip connections, is fabricated at a trusted foundry. The interposer, FEOL, and BEOL sections undergo final assembly in the trusted facility. In *3D integration*, both the FEOL and BEOL sections are fabricated by different foundries. The 3D assembly consists of vertically stacking the sections and inserting vertical interconnects called through-silicon vias (TSVs). Naturally with any Trojan countermeasure technique, there are tradeoffs with split manufacturing including higher manufacturing costs, increased area due to the interconnections, increased timing and power overheads, and higher temperatures in the middle tiers of the IC [6, 45, 46].

2.5.2 Detection of Trojans: Examples

2.5.2.1 Statistical Detection

Multiple excitation of rare occurrence (MERO) is a statistical form of Trojan detection [10]. MERO aims to maximize the probability of triggering Trojans during logic testing, while minimizing the number of test vectors required as compared with a weighted random pattern testing approach. It works by first detecting low-probability events at internal nodes and then creating a set of optimized test vectors that triggers each internal nodes to their rare logic value multiple times (e.g., $N > 1000$). It applies these vectors during testing in an effort to trigger any Trojans in the circuit. Recall that conventional testing does not scale to detect Trojans due to the exponential number of possible Trojan instances. Figure 2.22a shows the rare event ($abc = 011$) required to trigger a combinational Trojan, and Fig. 2.22b shows the rare occurrence ($ab = 10$) required to trigger a sequential Trojan. These two conditions will be identified by MERO and toggled many times in order to trigger and detect any Trojans in the circuits.

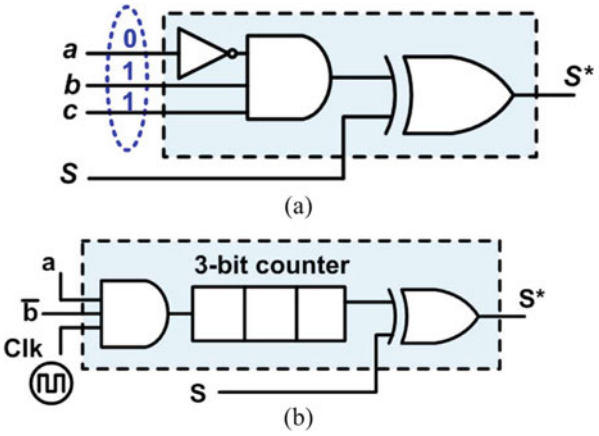

Fig. 2.22 Statistical logic testing of MERO for (**a**) combinational Trojan and (**b**) sequential Trojan [10]

2.5.2.2 Rare Event Removal

A methodology for identifying and removing rare nodes to increase Trojan detection is demonstrated in [31]. It is widely known that adversaries target low-controllable and low-observable nodes in which to insert Trojans, thus leading to challenges with Trojan detection methods. This technique analyzes the hardware design to identify nets with a transition probability less than a specific threshold value, which makes them ideal candidates for Trojans. The transition probability threshold value is modeled using a geometric distribution (GD) and estimates the number of clock cycles required for a node to transition. One or more dummy scan flip-flops (dSFF) are then inserted into the identified nodes to increase their probability of transitioning, all without the flip-flops affecting the timing or functionality of the design. The increased transitioning rate will result in a decrease of hard-to-activate nodes in the circuit (i.e., removal of the rare events). Removing these rare events is what allows for facilitating Trojan detection in the hardware, even if the gate-level netlist is not trusted. Small Trojans may be fully activated resulting in detection (via logic testing) through malfunctions and faulty circuit outputs. Large Trojans may be partially activated resulting in detection (via side-channel analysis) through measurable changes in signals such as transient power and path delay.

Figure 2.23a shows an original Trojan cone (i.e., logic gates connected to the input of a Trojan gate) consisting of all *AND* gates with the probability of generating a "1" ($1/256 = 0.0039$) at the Trojan gate ($T_g i$) being much less than generating a "0" ($255/256 = 0.9961$). Figure 2.23b shows a single dummy scan flip-flop *OR* gate inserted into the top net of the circuit. This single gate dramatically reduces the number of clock cycles required to transition the Trojan gate (T_{gi}), thereby increasing the transition probability of a "1" to be much greater than its original value ("1," $17/512 = 0.0332$; "0," $495/512 = 0.9668$). This increase in transition rate leads to an increased probability of Trojan detection.

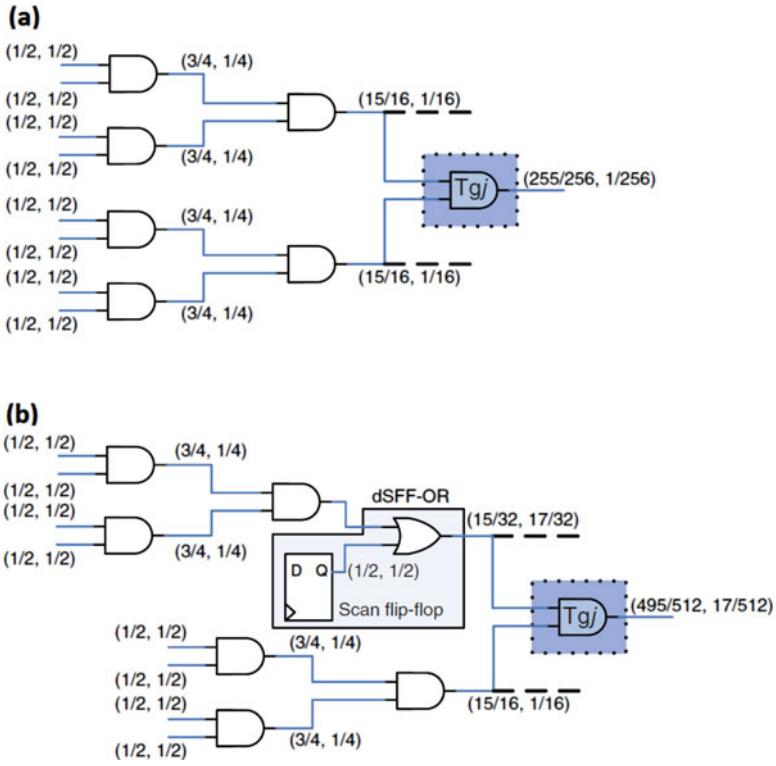

Fig. 2.23 Transition probability of the (**a**) original Trojan cone and (**b**) inclusion of a single dummy scan flip-flop [31]

2.5.3 Prevention of Trojans: Examples

2.5.3.1 Obfuscation

Key-based design obfuscation deters adversaries from inserting Trojans by transforming the circuit into another circuit that is functionally equivalent, but with the added benefits of security features to prevent hardware Trojans [8]. It allows a circuit to operate in two different modes (i.e., obfuscated mode and normal mode) and requires a sequential logic key to unlock correct circuit functionality. In the obfuscated mode (i.e., protected mode), the correct functionality and structural design of the circuit is obscured, resulting in incorrect behaviors being produced by the obfuscated circuit. In the normal mode, the correct behavior and functionality of the circuit is restored, although the design itself is still obscured from an adversary.

Transitioning from obfuscated mode to normal mode requires the correct logic key to be applied at the inputs during initial start-up of the circuit; otherwise the circuit will remain in its protected mode. This technique makes inserting

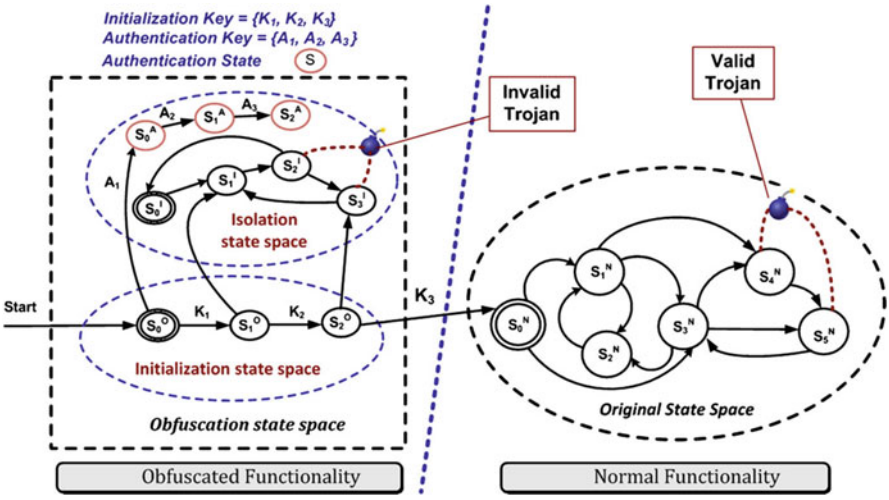

Fig. 2.24 Key-based obfuscation scheme for Trojan prevention and detection [8]

Trojans difficult for adversaries as the rare internal nodes are obfuscated (recall that adversaries insert Trojans into rare circuit nodes to prevent detection). If a Trojan is inserted, it is highly likely that it will be inserted into an isolated region, thus negating any effect of a triggered Trojan. This approach results in an increased prevention against hard-to-detect Trojans from being inserted into rare nodes, increased Trojan detection, as well as protection against IP piracy (i.e., IP theft via reverse engineering). It can prevent Trojans from being inserted by an adversary at an untrusted foundry or by untrusted EDA/CAD tools in IC design flows, albeit at a cost of higher overhead and the inability to prevent random Trojan insertions.

Figure 2.24 shows a state transition graph (STG) of a circuit. Upon power-up, the circuit starts in the obfuscated mode (i.e., state S_0^O). Only the correct sequential logic key (i.e., $K_1 \rightarrow K_2 \rightarrow K_3$) will transition the circuit into the normal mode (i.e., state S_0^N). Any incorrect key will result in the circuit transitioning into the isolation state space, where it will be trapped along with any invalid Trojans.

2.5.3.2 Hardware 3PIP

A design framework for a formal, yet computationally feasible, acquisition of trustworthy hardware 3PIP is demonstrated in the work of [21, 37]. This framework is called proof-carrying hardware intellectual property (PCHIP), and it is based on the concept of proof-carrying code (PCC). PCHIP focuses on the security and trust of the IP in the form of hardware description language (HDL) for FPGAs. Unlike many other approaches, it does not require a golden IC or model, nor does it require

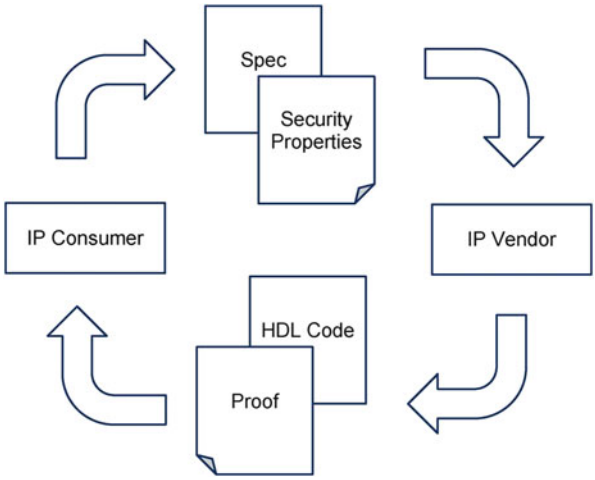

Fig. 2.25 Hardware intellectual property (IP) acquisition and delivery framework protocol [21]

a trusted 3PIP vendor. The IP consumer establishes a set of upfront security and trust properties with the IP vendor, which become integrated as a component in the entire design flow. These properties are temporal logic, meaning they are rule-based symbolic expressions instead of exact hardware functionality. The IP vendor creates a formal proof of these properties that is delivered with the 3PIP to the IP consumer. The IP consumer then validates the 3PIP against the agreed-upon security specifications. If the 3PIP fails verification, it means a security protocol has likely been violated, thus resulting in the prevention of malicious Trojans from being inserted into the IP consumer's FPGA design. Since an inadequate set of security specifications may have unknown vulnerabilities, this framework does not ensure complete coverage and should be used only in conjunction with other detection and prevention methods. Figure 2.25 shows the interaction cycle between the IP consumer and the IP Vendor.

2.5.4 Other Notable Trojan Detection and Prevention Methods

Similarly to the evolution of hardware Trojan designs attacks, Trojan detection and prevention mechanisms have also evolved in much the same manner. This section will introduce other notable and new Trojan detection and prevention techniques.

2.5.4.1 Voltage Inversion

A voltage inversion technique to ascertain malicious insertions (VITAMIN) in ICs was proposed by [2]. This technique utilizes an inverted voltage scheme to complement the voltage level of CMOS gates (e.g., changing an *AND* gate into a

NAND gate). It aims to detect the presence of hardware Trojans through enhancing the differences in the power profiles between a golden IC and a test IC, resulting in higher triggering frequencies for the rare nodes in the circuit. This approach was combined with a sustained vector technique to further strengthen its effectiveness.

2.5.4.2 Temperature Tracking

A run-time method for detecting hardware Trojans was introduced in [14]. The goal of this framework is to detect a deviation in the normal correlation between the thermal and power profiles within an IC. Abnormalities in this profile may indicate a Trojan activation. This approach consists of design-time, test-time, and run-time monitoring phases. It is low overhead as it exploits the existing internal thermal sensors on many FPGAs, SoCs, and other ICs. This approach enables online Trojan detection during the entire lifetime of the IC.

2.5.4.3 Split Fabrication

A secure split manufacturing methodology using vertical slit field effect transistors (VeSFET) inside of ICs is proposed by [46]. This approach uses the VeSFET's two-sided accessibility and 3D integration capability to hide transistors in a camouflaged 3D case. It allows two independent untrusted foundries to securely fabricate 2D and 3D ICs. If one foundry adds or moves transistors, the resulting crowbar current effect is detected by the other foundry. This design methodology prevents hardware Trojan insertion, reverse engineering, and IP piracy, as well as provides methods for Trojan detection capabilities.

2.5.4.4 FPGA Trust

Hardware Trojan attack prevention and detection methods for FPGAs are demonstrated in the works of [22]. This security and trust validation method, called adapted triple modular redundancy (ATMR), enables protection against Trojans that are inserted during device production at the foundry. It is adaptive since Trojans can be independent of the final design. ATMR is a combined approach consisting of logic testing and side-channel analysis, and it is designed to protect FPGAs from Trojans of various sizes, locations, and functionalities. This work also developed a Trojan attacks taxonomy and Trojan models that specifically target FPGAs. The taxonomy covers Trojans that alter the programmed state and I/O blocks of the FPGA, including Trojans inserted by the foundry that cause logical malfunctions and physical damage.

2.5.5 Comparisons of Various Trojan Defensive Methodologies

Detection and prevention of hardware Trojans attempt to solve the problem from two unique viewpoints. Many detection methods have been researched and the consensus is that detecting a small, quiet Trojan is very challenging, and the majority of the methods for detection are based on the use of a golden IC or golden model [6, 45]. Although the authors of [45] state that prevention may be better than detection, it is worth considering that a balance of the two methods is perhaps the best approach toward Trojan-free hardware.

Automatic test pattern generation (ATPG) is a method of circuit testing that is used to distinguish between the correct behavior of the design and the faulty behavior caused by defects. The goal of ATPG is 100% test coverage, and it is based upon the specification of the circuit. Structural testing may detect some Trojans since the number of nodes to test grows linearly with the number of inputs to the circuit. However, functional testing is much more complex as the number of nodes to test grows exponentially with the number of inputs to the circuit, thus making it almost impossible to detect all possible Trojans. Formal verification methods are algorithmic-based approaches that validate properties of the intended design. Several proposed methods for Trojan detection, including formal verification, identification and removal of suspicious signals, and equivalence checking, are detailed in [38, 43, 47].

Functional and structural testing, in the forms of logic testing, aim to activate unknown Trojans during the validation process and to propagate their effects to an observable output node [6]. Logic testing is a straightforward approach that works well for detecting ultrasmall Trojans. It is also robust in the presence of environmental noise and manufacturing variations. However, it is not scalable and quickly fails as the circuit or Trojan increases in size or complexity. The reason for the lack of scalability is difficulty in generating a complete set of test vectors due to the exponential increase in input combinations required to test all internal nodes and generate all outputs. Essentially, logic testing has difficulties exciting rare, low-controllable, and low-observable nodes. Also, logic testing cannot trigger externally activated Trojans, and it requires a golden IC or model for Trojan detection [6, 45].

Side-channel power analysis is a detection method predicated on the expected, albeit unknown magnitude, increase or decrease in the power overhead due to the inclusion of a Trojan. Common methods include static current analysis and transient current analysis. Static current analysis is a method of detection that relies on the inactivated Trojan causing noticeable change in current draw from the power supply due to the addition or modification of gates. Transient current analysis, on the other hand, allows for the detection of switching activity inside an activated Trojan [5, 6, 9, 45].

Side-channel analysis scales very well to detect large Trojans in both small and large circuits. It is effective for Trojans that do not cause observable malfunctions (e.g., data leakage), and it is easy to generate test vectors for Trojan detection.

However, this form of testing performs poorly in the presence of environmental noise and manufacturing variations and does not work well for detecting ultrasmall Trojans. This limitation is due to the Trojan's effects being masked by the noise in the IC. Side-channel detection is a noninvasive and passive form of Trojan detection, and its effectiveness depends on the signal-to-noise ratio (SNR) and the Trojan-to-circuit ratio (TCR). The SNR is important as the effect of the Trojan can be masked by system or environmental noise. The TCR is important as it is the measurement of Trojan size to circuit size. Detecting small Trojans in large circuits is a growing challenge as IC feature sizes are continually becoming smaller and the number of transistors continues to increase in hardware designs. As with logic testing, side-channel analysis also typically requires a golden IC or model for Trojan detection [6, 45].

Although logic testing and side-channel analysis can be performed numerous times and on as many ICs as desired, they too suffer from drawbacks including Trojans in rare nodes remaining inactivated, activated Trojans being masked by the presence of noise, or variations in the IC tolerances during the manufacturing process. Therefore, performing a combination of logic testing and side-channel analysis can provide the best coverage for detecting hardware Trojans. Figure 2.26 shows the comparison between these two noninvasive hardware Trojan detection methods.

Logic Testing (Functional / Structural)	Side-Channel Analysis
✓ Performs well for detecting *small* Trojans	✓ Performs well for detecting *large* Trojans
✓ Robust to process noise	✓ Easy to generate test patterns
✓ Typically requires no additional hardware overhead	✓ Typically requires no additional hardware overhead
✗ Hard to detect *large* Trojans	✗ Hard to detect *small* Trojans
✗ Hard to generate test patterns	✗ Susceptible to process noise
✗ Typically requires golden IC or golden model	✗ Typically requires golden IC or golden model

Fig. 2.26 Comparison of noninvasive hardware Trojan detection methods

2.6 Conclusion

Hardware Trojans are a valid threat to hardware security and trust of integrated circuits. These malicious modifications pose a safety and security risk to any electronic system as the underlying *hardware is no longer considered the root of all trust*. Trojans may remain undetected and inactivated for many years while waiting for a specific circumstance to trigger so they may deliver their nefarious payload. We have seen a plethora of Trojan attack designs and recognize the need for a solid defense posture. In order to combat the threat of Trojans, accurate models for attacks and countermeasures must be ensured. Only with these models will detection and prevention techniques be realized. Current work in this domain relies on tradeoffs between security and performance, and no defensive technique provides a 100% guarantee. Future work in this domain will help answer the question if golden-free models are a viable approach for hardware Trojan defense mechanisms.

References

1. S. Ali, D. Mukhopadhyay, R.S. Chakraborty, S. Bhunia, Multi-level attack: an emerging threat model for cryptographic hardware, in *Proceeding of the Design, Automation & Test in Europe (DATE) Conference Exhibition* (2011), pp. 1–4
2. M. Banga, M. Hsiao, VITAMIN: voltage inversion technique to ascertain malicious insertions in ICs, in *Proceeding of the IEEE International Workshop on Hardware-Oriented Security and Trust* (2009), pp. 104–107
3. C. Bao, D. Forte, A. Srivastava, On reverse engineering-based hardware Trojan detection. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **35**(1), 49–57 (2016)
4. G.T. Becker, F. Regazzoni, C. Paar, W.P. Burleson, Stealthy dopant-level hardware Trojans: extended version. J. Cryptogr. Eng. **4**(1), 1–13 (2014)
5. S. Bhunia, M. Abramovici, D. Agrawal, P. Bradley, M.S. Hsiao, J. Plusquellic, M. Tehranipoor, Protection against hardware Trojan attacks: towards a comprehensive solution. IEEE Design Test **30**(3), 6–17 (2013)
6. S. Bhunia, M.S. Hsiao, M. Banga, S. Narasimhan, Hardware Trojan attacks: threat analysis and countermeasures. Proc. IEEE **102**(8), 1229–1247 (2014)
7. B. Cha, S.K. Gupta, A resizing method to minimize effects of hardware Trojans, in *2014 IEEE 23rd Asian Test Symposium* (2014), pp. 192–199
8. R.S. Chakraborty, S. Bhunia, Security against hardware Trojan attacks using key-based design obfuscation. J. Electron. Test. (JETTA) Theory Appl. **27**(6), 767–785 (2011)
9. R.S. Chakraborty, S. Narasimhan, S. Bhunia, Hardware Trojan: threats and emerging solutions, in *IEEE International High Level Design Validation and Test Workshop* (2009), pp. 166–171
10. R.S. Chakraborty, F. Wolff, S. Paul, C. Papachristou, S. Bhunia, MERO: a statistical approach for hardware Trojan detection, in *Proceeding of the Cryptographic Hardware and Embedded Systems (CHES)* (2009), pp. 396–410
11. G.K. Contreras, A. Nahiyan, S. Bhunia, D. Forte, M. Tehranipoor, Security vulnerability analysis of design-for-test exploits for asset protection in SoCs, in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)* (2017), pp. 617–622

12. D. Du, S. Narasimhan, R.S. Chakraborty, S. Bhunia, Self-referencing: a scalable side-channel approach for hardware Trojan detection, in *Proceeding of the Cryptographic Hardware and Embedded Systems (CHES)* (2010), pp. 173–187
13. C. Dunbar, G. Qu, Designing trusted embedded systems from finite state machines. ACM Trans. Embed. Comput. Syst. **13**(5s), Article 153 (2014)
14. D. Forte, C. Bao, A. Srivastava, Temperature tracking: an innovative run-time approach for hardware Trojan detection, in *Proceedings of the 2013 IEEE/ACM International Conference on Computer-Aided Design, ICCAD* (2013), pp. 532–539
15. Y. Jin, N. Kupp, CSAW 2008 team report (Yale University). CSAW embedded system challenge (2008). [Online], Available: http://isis.poly.edu/vikram/yale.pdf
16. Y. Kaiyuan, M. Hicks, Q. Dong, T. Austin, D. Sylvester, A2: analog malicious hardware, in *2016 IEEE Symposium on Security and Privacy (SP)* (2016)
17. R. Karri, J. Rajendran, K. Rosenfeld, M. Tehranipoor, Trustworthy hardware: identifying and classifying hardware Trojans. IEEE Comput. **43**(10), 39–46 (2010)
18. S.T. King, J. Tucek, A. Cozzie, C. Grier, W. Jiang, Y. Zhou, Designing and implementing malicious hardware, in *Proceeding of the 1st USENIX Workshop Large-Scale Exploits Emergent Threats (LEET)* (2008)
19. L. Lin, W. Burleson, C. Paar, MOLES: malicious off-chip leakage enabled by side-channels, in *Proceedings International Conference on Computer-Aided Design (ICCAD)* (2009), pp. 117–122
20. Y. Liu, Y. Jin, A. Nosratinia, Y. Makris, Silicon demonstration of hardware Trojan design and detection in wireless cryptographic ICs. IEEE Trans. Very Large Scale Integr. VLSI Syst. **25**(4), 1506–1519 (2017)
21. E. Love, Y. Jin, Y. Makris, Proof-carrying hardware intellectual property: a pathway to trusted module acquisition. IEEE Trans. Inf. Forensics Secur. **7**(1), 25–40 (2012)
22. S. Mal-Sarkar, R. Karam, S. Narasimhan, A. Ghosh, A. Krishna, S. Bhunia, Design and validation for FPGA trust under hardware Trojan attacks. IEEE Trans. Multi-Scale Comput. Syst. **2**(3), 186–198 (2016)
23. S. Narasimhan, X. Wang, D. Du, R.S. Chakraborty, S. Bhunia, TeSR: a robust temporal self-referencing approach for hardware Trojan detection, in *Proceeding of the IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)* (2011), pp. 71–74
24. S. Narasimhan, W. Yueh, X. Wang, S. Mukhopadhyay, S. Bhunia, Improving IC security against Trojan attacks through integration of security monitors. IEEE Des. Test Comput. **29**(5), 37–46 (2012)
25. S. Narasimhan, D. Du, R.S. Chakraborty, S. Paul, F.G. Wolff, C.A. Papachristou, K. Roy, S. Bhunia, Hardware Trojan detection by multiple-parameter side-channel analysis. IEEE Trans. Comput. **62**(11), 2183–2195 (2013)
26. M. Potkonjak, Synthesis of trustable ICs using untrusted CAD tools, in *Proceeding of the Design Automation Conference* (2010), pp. 633–634
27. J. Rajendran, A.K. Kanuparthi, M. Zahran, S.K. Addepalli, G. Ormazabal, R. Karri, Securing processors against insider attacks: a circuit-microarchitecture co-design approach. IEEE Des. Test **30**(2), 35–44 (2013)
28. T. Reece, D.B. Limbrick, W.H. Robinson, Design comparison to identify malicious hardware in external intellectual property, in *Proceeding of the IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, Changsha (2011), pp. 639–646
29. M. Rostami, F. Koushanfar, J. Rajendran, R. Karri, Hardware security: threat models and metrics, in *Proceedings of the International Conference on Computer-Aided Design (ICCAD'13)* (IEEE Press, Piscataway, 2013), pp. 819–823
30. J. Roy, F. Koushanfar, I. Markov, EPIC: ending piracy of integrated circuits. IEEE Comput. **43**(10), 30–38 (2010)
31. H. Salmani, M. Tehranipoor, J. Plusquellic, A novel technique for improving hardware Trojan detection and reducing Trojan activation time. IEEE Trans. Very Large Scale Integr. VLSI Syst. **20**(1), 112–125 (2012)

32. H. Salmani, M. Tehranipoor, R. Karri, On design vulnerability analysis and trust benchmark development, in *IEEE International Conference on Computer Design (ICCD)* (2013)
33. M. Sanchita, A. Krishna, A. Ghosh, S. Bhunia, Hardware Trojan attacks in FPGA devices: threat analysis and effective counter measures, in *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI* (2014), pp. 287–292
34. B. Shakya, T. He, H. Salmani, D. Forte, S. Bhunia, M. Tehranipoor, Benchmarking of hardware Trojans and maliciously affected circuits. J. Hardw. Syst. Secur. (HaSS) **1**(1), 85–102 (2017)
35. Y. Shiyanovskii, F. Wolff, A. Rajendran, C. Papachristou, D. Weyer, W. Clay, Process reliability based Trojans through NBTI and HCI effects, in *Proceeding of the NASA/ESA Conference on Adaptive Hardware and Systems* (2010), pp. 215–222
36. M. Tehranipoor, F. Koushanfar, A survey of hardware Trojan taxonomy and detections. IEEE Des. Test Comput. **27**(1), 10–25 (2010)
37. M. Tehranipoor, C. Wang, *Introduction to Hardware Security and Trust* (Springer, New York, 2012)
38. M. Tehranipoor, H. Salmani, X. Zhang, *Integrated Circuit Authentication* (Springer, Cham, 2014)
39. M. Tehranipoor, U. Guin, D. Forte, *Counterfeit Integrated Circuits: Detection and Avoidance* (Springer, Cham, 2015)
40. R. Torrance, D. James, The state-of-the-art in semiconductor reverse engineering, in *IEEE/ACM Design Automation Conference* (2011), pp. 333–338
41. TrustHub. https://www.trust-hub.org/index.php
42. N.G. Tsoutsos, M. Maniatakos, Fabrication attacks: zero-overhead malicious modifications enabling modern microprocessor privilege escalation. IEEE Trans. Emerg. Top. Comput. **2**(1), 81–93 (2014)
43. X. Wang, M. Tehranipoor, J. Plusquellic, Detecting malicious inclusions in secure hardware: challenges and solutions, in *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)* (2008)
44. X. Wang, S. Narasimhan, A. Krishna, T. Mal-Sarkar, S. Bhunia, Software exploitable hardware Trojan attacks in embedded processor, in *Proceedings of the IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)* (2012), pp. 55–58
45. K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, M. Tehranipoor, Hardware Trojans: lessons learned after one decade of research. ACM Trans. Des. Autom. Electron. Syst. **22**(1), 6:1–6:23 (2016)
46. P.L. Yang, M. Marek-Sadowska, Making split-fabrication more secure, in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Austin (2016), pp. 1–8
47. X. Zhang, M. Tehranipoor, Case study: detecting hardware Trojans in third-party digital IP cores. IEEE Int. Symp. Hardw.-Oriented Secur. Trust **22**(1), 67–70 (2011)
48. Y. Zheng, S. Yang, S. Bhunia, SeMIA: self-similarity-based IC integrity analysis. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **35**(1), 37–48 (2016)