Assessing and Detecting Malicious Hardware in Integrated Circuits

By

Trey Reece

Dissertation
Submitted to the Faculty of the
Graduate School of Vanderbilt University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

December, 2014
Nashville, TN

ELECTRICAL ENGINEERING

Assessing and Detecting Malicious Hardware in Integrated Circuits

Trey Reece

Dissertation under the direction of Professor William H. Robinson

System security often focuses on the software, causing hardware security to be overlooked. Such oversight allows for attacks that can completely undermine the use of hardware as the root of trust. During the design of an integrated circuit, there are several opportunities for adversaries to make malicious modifications or insertions to a design. These attacks, known as hardware Trojans, can have catastrophic effects on a circuit if left undetected. This dissertation addresses Trojan impact, and proposes two low-cost detection methods for hardware Trojans inserted at different points in the production pipeline, namely: (1) Trojans hidden within third-party intellectual property (IP) licensed from a vendor, and (2) Trojans inserted during the fabrication steps taking place at external fabrication plants.

Determining whether third-party IP does only its intended function and nothing else is a major challenge. Through comparison of two similar but untrusted designs using design unrolling and Boolean satisfiability, functional differences can be identified for all possible input combinations within a window of time. This technique was tested on multiple Trojan benchmarks and demonstrated effective and accurate detectability.

Process variation poses the greatest obstacle to detecting Trojans in chips. In order to detect Trojans inserted during fabrication, a digital model is created. This model can then be trained using the fabricated chip to account for different parameters, such as process variation. The parameters of the trained model can be used to identify suspicious areas of the chip. Furthermore, this process does not require expensive test equipment, nor does it require a costly golden, trusted chip for comparison.

Acknowledgements

First, I would like to thank my family for their continued support. It's been a long trek, and they never stopped believing in me. This dissertation would not be what it is now without their constant help. I would also like to thank my advisor, Dr. William H. Robinson, for putting up with me over the years. His steadfast support has been critical in finishing this work, and for me in finishing my degree. Further thanks go to Katy Clark, for all of her help with proofreading and editing. Special thanks is also deserved for Dr. Bharat Bhuva, who has given me a huge amount of assistance ever since I darkened his door as an undergraduate. I also thank those in the JPAC group for their support and friendship. Finally, I would like to thank the National Science Foundation for financial support in performing my research.

TABLE OF CONTENTS

List of Figures

List of Tables

List of Abbreviations

AES	Advanced Encryption Standard
ALU	Arithmetic Logic Unit
ATPG	Automated Test Pattern Generation
DFT	Design for Test
EDA	Electronic Design Automation
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
IC	Integrated Circuit
IP	Intellectual Property
JTAG	Joint Test Action Group
MISR	Multiple-Input Signal Register
NMOS	n-type Metal-Oxide-Semiconductor
PMOS	p-type Metal-Oxide-Semiconductor
SMV	Symbolic Model Verifier
TMR	Triple Modular Redundancy
UART	Universal Asynchronous Retriever/Transmitter
VHDL	VHSIC Hardware Description Language

Chapter 1

Introduction

The continued advancement of modern integrated circuits has played a significant part in technological advancements over the past few decades. Unfortunately, as technology advances, opportunities to abuse or subvert this technology also become available. To most designers, one of the advantages to implementing a design in hardware instead of as a software implementation is the secure nature of hardware. The assumption that hardware is secure while software can be attacked, is prevalent. Unfortunately, this is a false assumption, created due to a lack of security awareness with increasingly complicated circuits [2].

Modern integrated circuits can be incredibly complicated, incorporating billions of transistors, and making use of resources from hundreds of sources. Many of these external resources can probably be trusted, such as the Cadence or Synopsys tool suites as they are the standard tools used for chip design. Unfortunately, other external resources cannot, and most likely should not, be trusted. There is no guarantee that a design sent overseas for fabrication will not have malicious changes in the final chip. It is important to identify threats and possible risks, and determine what is trustworthy and what threats need to be mitigated.

An unaddressed threat represents a possible vector for an adversary to insert an attack into a fabricated chip. Such attacks, dubbed "hardware Trojans", can have a wide variety of payloads for an even wider variety of malicious purposes [3]. For instance, an attacker could easily modify a cryptographic circuit to leak a key, set a critical circuit to self-destruct on a specific signal, or even covertly monitor and control a microcontroller remotely [4]. As long as the circuit appears to operate correctly under test conditions, such an attack would likely remain unnoticed.

While there are many points during production during which an adversary could influence a circuit, the possible damage that can be inflicted is maximized during the design and fabrication stages. Prior to the design stage, most malicious input would be limited to intentional sabotage of the overall design. Similarly, after the fabrication stage the attacker would generally need to rely on either intentional sabotage, or physically swapping the fabricated design with a subverted chip. However, during the design and fabrication stages, the damage is much more subtle.

During the design stage, an adversary could hide malicious functionality inside of a third-party design block, masking it as an Intellectual Property (IP) watermark. Such functionality would remain in the circuit even after fabrication, and could be a veritable landmine whenever an attacker needed to call upon it. During the fabrication stage, an attacker at a fabrication plant could have free reign to modify the circuit, and the original designer would likely never know. The only certain way to identify such an attack would be to tear the fabricated chip apart with destructive imaging techniques, and digitally compare the designs. While such reverse engineering may be effective at finding Trojans, despite the small size of modern fabrication technologies [5], this process is much too expensive for a chip that most likely does not have a hidden Trojan. Taking the costs into account, very few Trojans will be identified by imaging techniques. Effective methods that provide a more cost-effective alternative must be provided.

1.1 Research Contributions

The contributions of this research can be summarized as follows:

- **Development and testing of a technique to identify Trojans hidden within third-party intellectual property** This technique samples IP blocks from multiple vendors and uses them as validating circuits for each other. This aids designers in preventing inclusion of malicious functionality in both licensed and unlicensed third-party design blocks.

- **Assessment of the impact of malicious modifications on fabricated chips in terms of area and power consumption.** The primary benefit from this research comes from the characterization of what amount of impact can arise from a Trojan, and therefore how small of an impact must be detectable for counter-Trojan techniques.

- **A functional modelling method for identifying Trojans hidden within fabricated circuits without need of a physical golden copy.** This method revolves around adapting a known model of a circuit to match power and timing responses for an unknown physical circuit. This adapted model can then be examined after testing to identify signs of possible Trojan circuitry hidden in the unknown circuit. This allows chip designers to identify suspicious chips in a lot even without access to known safe copies of chips, or "golden" chips.

1.2 Organization of Dissertation

The dissertation is organized as follows:

The first portion of the dissertation, covering Chapters 2 through 5, covers key background information regarding malicious hardware and existing research on the topics.

- Chapter 2 describes the definitions with which risk and trust are measured and verified.
- Chapter 3 describes the background of hardware Trojans, and provides details for their structure and classification.
- Chapter 4 describes the process from the conception of a specification of an integrated circuit (IC) to the testing of the fabricated design, and identifies many possible attack vectors during this process.
- Chapter 5 explains how many of the threatened steps during production can be secured.

The second portion of this dissertation, covering chapters 6 through 9, describe the research performed, and techniques developed in this dissertation.

- Chapter 6 describes the technique developed within this dissertation to identify Trojan functionality hidden within third-party intellectual property, and its results.

- Chapter 7 reports the research and results in exploring the impact of Trojan functionality on host circuitry in terms of area and power.
- Chapter 8 describes the modelling technique developed in order to identify Trojans hidden within fabricated circuitry during the fabrication process.
- Finally, Chapter 9 summarizes this work, and provides extensions for future study.

Chapter 2

Risk Management

One of the prevailing assumptions in the area of security is the false belief that hardware implementations are secure. Due to the sheer number of attacks implemented through software, a mistaken belief has arisen that attacks will only come through software. While it is certainly true that the vast majority of attacks on a design might be through software, this does not preclude the possibility that an attack will also be implemented inside the hardware. In a complex design, many assumptions are made about the security of each step, and at each abstraction level. Everything at higher abstraction levels is built with the assumption that the lower abstraction levels operate securely (see Figure 2-1). If an operating system is secure, but operating on a compromised digital design, then can we call this implementation secure? Is it even possible to build a secure implementation upon an insecure base?

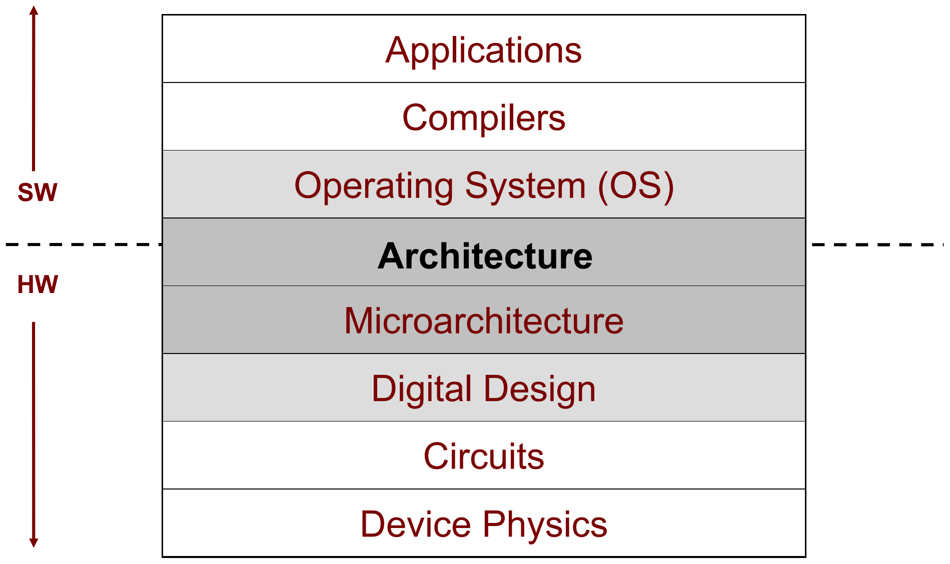

Figure 2-1: Levels of system abstraction for integrated circuits, classified into hardware (HW) stages and software (SW) stages.

2.1 What can we Trust?

When determining the security of a design, the first step is to identify clearly what types of steps can be trusted, and what types cannot. This determination might change depending upon types of circuits and their implementations, but for this dissertation, one easy rule will be implemented: in-house work can be trusted, while out-of-house work cannot.

2.1.1 In-House

A simplifying assumption made for the purpose of this work is that that all in-house work can be considered "trusted". The point of this assumption is to clearly define attack vectors in order to most effectively block possible attacks. This allows a designer to guarantee that every possible step in a design is covered from attacks. Under no circumstances does this mean that there are no security leaks, attempted sabotage, theft, or other problems within an organization. In fact, many organizations are rife with this type of in-house threat. However, there are many effective methods to resolve these threats that can be put into place [6]. It is difficult to secretly sabotage a design if all changes to a digital design are tracked and logged, with significant oversight on all changes. To put it simply: in-house work has its own process of verification that acts completely separately from other types of verification.

2.1.2 Out-of-House

We further assume that all work performed outside of an organization is suspect. Any production step in which a design is modified by, or in the care of, an outside source can represent a possible vector for an attack. For each step, it is important to identify what attacks might be made by a third party during this opportunity, and determine methods of either preventing or identifying such attacks. For example, a medical device company designing a pacemaker might license a wireless controller block from a vendor marketing third-party intellectual property (IP). It could be disastrous if this controller had malicious functionality hidden by the designer. In such a situation, it is critical to identify the risk posed by incorporating this untrusted block in a design.

Another important aspect of the environment to note is that each step represents a different type of risk, which needs to be assessed by the designer. In the previous example of a pacemaker, there is a high level of risk for the design due to the disastrous effects of their device failing. There would be less need to incorporate such strict security for a wireless controller in a car alarm, as the cost of failure is lower. Failing car-alarms are much less likely to cause death than failing pacemakers.

2.1.2.1 Vulnerabilities in the Supply Chain

One reason that out-of-house resources are assumed untrusted in this dissertation, regardless of what is purported to be supplied, is because tracking the source of an external resource can be long and convoluted. This has been a significant issue with defense contractors in the past few years, with regards to actual physical chips purchased from reputable vendors or resellers.

Consider several examples. First, a fiasco involving the United States Navy was made public in 2010, when a company called VisionTech was charged with selling over 59,000 microchips which contained hidden kill-switch functionality [7]. This functionality would allow an attacker to remotely disable whatever was running on these chips, including missiles, communication equipment, and other military vehicles. For years, this company had been importing counterfeit chips from China and marketing them to defense companies as military grade microchips. The list of these companies includes BAE systems, which provides Identification Friend-or-Foe (IFF) systems to the US Navy, and Raytheon Missile systems, for incorporation on F-16 fighter planes.

Second, unfortunately VisionTech is not the only reseller to buy cheap microchips from overseas and sell them domestically. Another similar example of corruption in the supply-chain transpired in 2005 when United Aircraft and Electronics, a company in which the operator sold false certification of aircraft parts, and was sentenced to 188 months in prison [8].

Third, an example from 2002 is the case of United Space Alliance, a company which bid and received a $24 million contract with NASA to supply military grade 8086 microprocessors for use with the space-shuttle computers. This company then proceeded to purchase used computers through eBay and pull commercial-grade 8086 processors from the motherboards [8]. Commercial-grade chips would almost certainly have difficulties operating in the adverse environments required by the space shuttle computers.

Unless it is possible to completely track the life of a resource, that resource should be considered suspect. Since verification of an external resource is generally a simpler task than a full forensic investigation of the history of a resource, verification is the preferred method of determining whether something can be considered trusted.

2.2 How do we Trust?

Although one can make the argument that something can be made 100% trustworthy by simply doing everything in-house, this is not an option for most organizations. One of the primary reasons for outsourcing individual steps in a production path is to save time and money. If avoiding outsourcing causes a design to become too expensive to compete on an open market, then there is little point in creating the design.

What is necessary in this case is a method of taking an untrusted input, be it a circuit, a digital design, or a cell library, and verify that this untrusted input has no hidden malicious functionality. Depending on the untrusted step, this process will be different.

2.2.1 Verification

The first step when exploring the problem of verification is to explicitly define what is meant by the word 'verification'. For some groups, verification of a design means the manual or logical proof that a design functions as intended for all possible inputs and states. This is often used with digital designs when it is possible to form a mathematical proof. However, according to Godefroid [9]:

- *To verify means to mathematically prove that a system meets its correctness requirements. Verification is thus the tool that enables the designer to be confident that the formal description of the system he/she has obtained does indeed satisfy the problem requirements.*

Under this definition, only when every possible input situation is proven to act as intended, the system is verified. However, when dealing with something other than a computer system or a simulated design, such a mathematical proof can become impossible. This also can happen as the designs grow in size and complexity. In many cases, it is useful to consider an alternate version of verification:

- *as a means to test and show that the design works* well enough *and will act as intended through all* expected *situations.*

Even though both of these definitions of verification have the same general purpose, the actual process involved with each type of verification varies significantly. This difference is predominantly due to problem known as the state explosion dilemma, which is reviewed below.

2.2.2 The State Explosion Dilemma

Most modern designs are too complicated to allow for exhaustive testing of every possible input. This is due to the rate at which the number of possible tests increase. For instance, a simple circuit with no internal states and four inputs has a total of 2^4, or 16 possible patterns to test. If the number of inputs were to be increased to 10, there would be 2^{10}, or 1024 possible test patterns. This increase is exponential, growing increasingly quickly with each additional input. To make matters worse, most modern designs are not stateless; they incorporate possible internal states affecting the response for each input. During an exhaustive test, each state would need to be tested with every possible input. The total number of test patterns can be explained with the relation $2^{(n+m)}$ where n is the number of input bits, and m is the number of state bits (i.e., flip-flops or latches in a circuit). As such, the work necessary for a comprehensive test increases exponentially with a linear increase in the size of a problem. This quickly becomes too large to test practically, even for small circuits. For example, a circuit containing 20 inputs and 60 state bits would require over 1.21×10^{24} tests if every possible state were to be tested. At 2.4 GHz with one test per cycle, this would take almost 16 million years to finish. Even if this work were to be parallelized and split among thousands of processors, it would still take years. Clearly, another method of verifying the validity of a design is necessary.

2.2.2.1 Addressing the problem

While the state explosion dilemma can be easily defined, it is difficult to address as a problem. There has been a great deal of research on effective methods to reduce the cost of such thorough tests. In many cases, these workarounds focus on a specific test condition that allows the problem to be simplified, reducing the computational cost of testing. By far the most common method of simplifying the problem involves converting the problem into state machines, and then comparing them for equivalence [10, 11]. Although this is faster than directly testing every input, this process can be quite taxing due to the cost of testing equivalence, as well as the cost of building the state machines themselves (although this aspect of the cost has decreased with the availability of inexpensive memory, and is no longer a significant problem).

For example, Coudert et al. [12] describes a methodology for comparing every possible input between a circuit design specification and a circuit realization. In this situation, there are two identical implementations of the same circuit that are implemented at different abstraction levels. These circuits are represented with deterministic Moore machines. Equivalence is then proven by symbolically executing both of the two circuits simultaneously using binary decision diagrams, removing the need to build a state machine to show equivalence. This research was followed up by a more formal technique [13] which optimizes the testing using computation tree logic instead of symbolic breadth-first execution.

A similar solution is proposed by Probst et al. [14] wherein model-checking for delay-insensitive VLSI systems is performed by removing all states from the equation and, instead, considers everything in terms of processes. The logic behind this transformation relies upon the interpretation that behavioral machines describe the branches and recurrences present within processes. By performing this transformation, the inherent difficulty with the state explosion dilemma is sidestepped, as there are no longer any states in the problem. Regardless, all of these methods focus on reducing the complexity of the problem of testing every possible input when matching two designs.

2.2.3 State Machines or Finite State Automata

When dealing with a hard verification of a computer system or a simulated design, it is often possible to adapt the representation of the design such that it can be explained in terms of Finite State Machines or Finite State Automata. Once a design is expressed in terms of a state machine, the plethora of research conducted on state machines are available to solve the problem. For instance, if within a state machine representation of a system we can show that any state can traverse to any other state, then we can mathematically prove that the system represented will never permanently stall. As a more complicated example, if we can show that two state machines are functionally equivalent, then we can prove that the two systems represented by these state machines are equivalent. If one of these systems is a set of design requirements, and the other system is a complete design, then this process proves that a design meets the necessary design requirements. Considering how proving equivalence between two state machines is an NP-Hard problem [15], there have been many optimizations proposed for this process.

One such technique was proposed by Burch et al. [16] through the use of a symbolic representation of a state machine, as opposed to an explicit one. This involves a combination of binary decision diagrams [17], and an abstract dialect of Modal μ-calculus (Mu-Calculus). This reduces the state explosion problem to a solution of a Mu-Calculus formula, which is an explored and highly optimized calculation. Other possible optimizations can take on significantly different forms. Some rely on changing the representation of the problem (e.g. Probst et al. [14] and Burch et al. [16]) while others rely on taking advantage of the concurrency present in certain state machines [18, 19, 14]. It is important to note that all of these methods use the first, most rigorous, definition of verification introduced; i.e., thorough verification requires 100% coverage of all possible inputs.

2.2.4 Design Verification

Instead of a comprehensive test over every possible input, another possible approach is to reduce testing to a specific set of patterns called a test vector. The goal of these tests is to assess specific conditions for errors, without testing every input. These tests are applied on designs when the only verification necessary is proof that the circuit works, as opposed to a rigorous test to show perfection.

This type of simplification greatly reduces the amount of work, as it is only necessary to test enough of a circuit to meet the desired verification requirements. As an example, even if it cannot be proven that a system will *never* have an error, it could be possible to prove that the system will not have an error within X years (where X is a suitable length of time based on the system in question). If the hardware itself will fail long before any logical failure might occur, then it does not matter whether or not the logical failure exists. As an extremely simple example, suppose the existence of a circuit that has 5 inputs, but the circuit supplying these inputs will only supply either all '0's or all '1's. In this situation, there is little reason to test all possible combinations of '0's and '1's, as they will not occur during actual use. Unless this circuit is connected to a system that will provide an invalid input, these extra states are largely unnecessary and expensive to test. The only point in testing these extra inputs would be if there was some reason to suspect the circuit supplying the inputs was not trustworthy.

Another advantage to this type of testing is that it can be performed without having a digital or symbolic representation of the circuit. Through bypassing such representation, this type of testing can be used as a method of verifying physical systems that cannot be mathematically analyzed. For example, a test can be performed on a fabricated design to identify all possible faults within the design. For such a design, the purpose of this testing is to show that the physical implementation was fabricated without error and that all components within the design are functioning as intended. In order to find an effective set of tests, it is necessary to rely on an efficient algorithm to identify test vectors to run. This is generally called Automatic Test Pattern Generation (ATPG), and there are many commercially available programs that can perform this function (such as TetraMAX ATPG from Synopsys).

2.2.4.1 Automatic Test Pattern Generation

Despite the first impression of an ATPG test vector as random test, an effective ATPG algorithm cannot simply introduce a randomized selection of inputs to the circuit. The difficulty of identifying failures can vary greatly depending upon the location of the failure, as well as the type. Even worse, some faults are impossible to test, significantly complicating the task. To handle these complications, most ATPG algorithms are designed to solve for input vectors from the list of failures to test. For example, one of the earliest examples of an ATPG algorithm was published in 1966 by Roth [20] describing a method for generating a logical test to verify a given failure: the 'D-algorithm'. The purpose of this algorithm was to produce a test for a given failure, so long as such a test exists. Following this algorithm, one can proceed through the list of all possible faults and produce the tests necessary to assess failures at as many points as necessary.

Depending on the application, there are many different algorithms used for this type of testing. For example, a test pattern might focus on identifying all stuck-at-faults, where each node in a circuit is checked to make sure that it is not stuck at either a logical '1' or a '0'. This generally creates a total of $2n$ test cases, where n is the total number of nodes in a circuit at which a fault could occur. This can be further optimized for reduction in test pattern size, such as by Tsai et al. [21] which describes an optimization to classic ATPG fault-detection by forming clusters of test vectors that have a deterministically high likelihood of identifying faults in a circuit.

While effective for many failures, the stuck-at-fault method is yet another way of focusing on a smaller subset of the problem. These algorithms assume that all failures are permanent, and that failures are independent of each other. If a failure was to appear sporadically, then such a test would not account for it. Additionally, if a state device such as a flip-flop or latch was malfunctioning, then it might also produce unreliable errors. For this reason, it is necessary to combine multiple methods to effectively reduce the likelihood of a failure being untested.

One reason why many failures are difficult to identify is because of the way the circuits are designed. For instance, a circuit with Triple Modular Redundancy (TMR) is nearly impossible to verify as initially designed. A circuit with TMR essentially has 3 identical copies of itself, along with a majority-voting circuit for the output. With the exception of failures in the voting hardware, no single-failures will propagate to the output. The only way an error would be visible at the output is if two failures occurred for the same input for two copies of the circuit. If the only way to determine failures was through analysis of the intended output, then this circuit would be nearly impossible to verify. However, if we take these issues into account at design, then we could find ways around this problem. This process is known as Design for Test (DFT) [22, 23]. If the designers intentionally add extra outputs to the circuit to circumvent this redundancy, then it suddenly becomes possible to verify the circuit. By accounting for testing during the design phase of a circuit, it becomes significantly easier to identify any failures in a circuit. Examples of such DFT techniques include scan chains [24, 25, 26] and built in self-test [27, 28]. An incredible amount of time and money has been invested in researching methods of optimizing these tests, due to the increasing cost of fixing such failures as the production process progresses.

Chapter 3

Hardware Trojans

One of the most insidious methods of attacking a circuit is by modifying its hardware in a malicious way. To put it simply, a Hardware Trojan is created by discreetly inserting hidden functionality into a Hardware Design. This insertion can occur at any stage in a production path, and could have devastating effects on the final design. Such Trojans can have a variety of functionality, ranging from denial-of-service functionality that gives designs a controllable kill switch, to hidden data leaks that can leak sensitive information.

3.1 Background and Risk

One of the earliest papers covering the concept of Hardware Trojans was published by a group of researchers at the University of Illinois at Urbana-Champaign [4]. This research included the design and testing of a variant of the Aeroflex Gaisler LEON 3 [29] processor, dubbed the Illinois Malicious Processor (IMP). This IMP was a fully functional version of the LEON 3 that would work just fine in almost all circumstances, with the sole exception of one trigger: the receipt of a specially crafted corrupt network packet. Triggering this functionality would then switch the processor to a new shadow mode which the processor would accept and perform commands sent over the network, allowing an attacker to compromise and hijack a system running on this processor, regardless of any security measures in the software. Additionally, this modification only required the insertion of 1341 gates to the original circuit, which initially contained over 1 million.. Detecting such an insertion representing 0.1% of the circuit poses a significant problem. Even in much smaller circuits, the percent impact of hardware Trojans on the total area of a circuit is less than 0.5% [30].

Unfortunately, this is not merely an academic threat with no bearings on the real world. Hardware Trojans have been suspected in various applications in recent history. For instance, the 2007 Israeli Airstrike on a Syrian nuclear reactor dubbed Operation Orchard, is suspected to have been facilitated through hidden kill-switch functionality in the Syrian radar systems [3]. This functionality was then thought to be used to disable the Syrian radars for the short duration of the attack.

Another recent backdoor made headlines when military grade field-programmable gate array (FPGA) chips from Actel were accused of containing backdoor functionality [31]. These researchers found that there was functionality similar to an administrative debug built into the JTAG functionality of the Actel/Microsemi ProA-SIC3 A3P250, a widely-used chip with high security specifications. Using this backdoor, researchers were able to extract all configuration data from the chip, and make whatever modifications they wanted to the programmable FPGA cells. Considering how this chip was advertised as providing "*the most impenetrable security for programmable logic designs*", this was a surprising turn of events. While this backdoor could have been due to debugging or testing functionality that was not disabled, it also acts effectively as a hardware Trojan.

3.2 Insertion

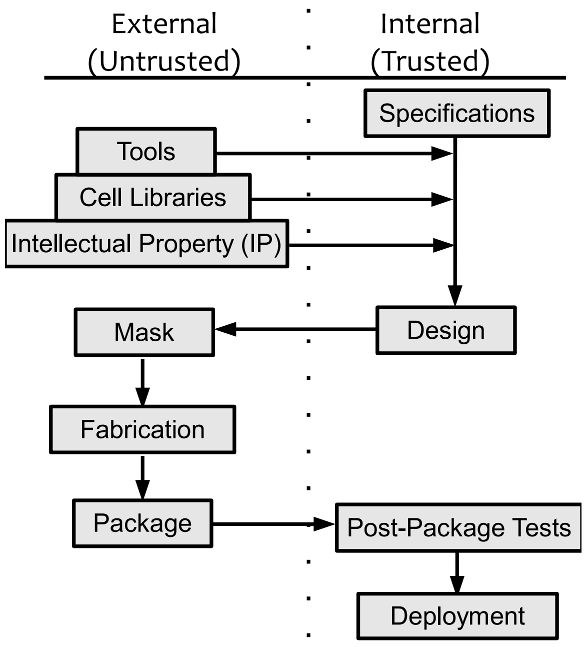

Figure 3-1: Example Circuit Production Path Providing Trust Estimates of Individual Steps

Figure 3-1 represents an example production path representing the process necessary to create a single microchip, from concept to packaging. For a normal typical, many of these steps are outsourced to a third party and therefore represent a vector for possible attack on a design. Unfortunately, it is infeasible for most companies to attempt to move every single one of these untrusted steps to in-house production, as it would be both prohibitively expensive, and incredibly inefficient.

This is especially the case when it comes to the steps involving fabrication, due to the skyrocketing price of modern fabrication plants. For the vast majority of companies, the only possible way to stay competitive in today's market is to rely on overseas fabrication plants. One of these fabrication plants could easily make changes to the layout before fabrication, and the purchasing company would have no way of identifying such a modification, short of stripping the packaging off and submitting it to expensive and destructive imaging processes. At this time, there are a number of researchers working to develop inexpensive methods of identifying such Trojans, specifically in a nondestructive process. These works are further described in Chapter 5.

Another easy point of insertion for a Trojan is through the inclusion of third-party Intellectual Property (IP), licensed during the assembly of the design. There are a vast number of companies marketing IP for use in designs, with little oversight to verify that these designs actually do what they are supposed to do, and nothing else. Also it would be prohibitively expensive for a company to design everything from scratch, and could also increase the possibility of errors.

3.3 Taxonomy

Hardware Trojans can come in all shapes and sizes, and with a large variety when it comes to functionality. Since the definition of a Hardware Trojan is broadly "A malicious modification made in the hardware," all types of modifications are included. This could include attacks such as a wire thinned at fabrication for intentional reduced reliability, to a complicated structure leaking sensitive data from a high-performance cryptographic block. A detailed taxonomy was proposed by Karri et al. [32], which breaks Trojans down according to five attributes. The figure is replicated below in Fig. 3-2

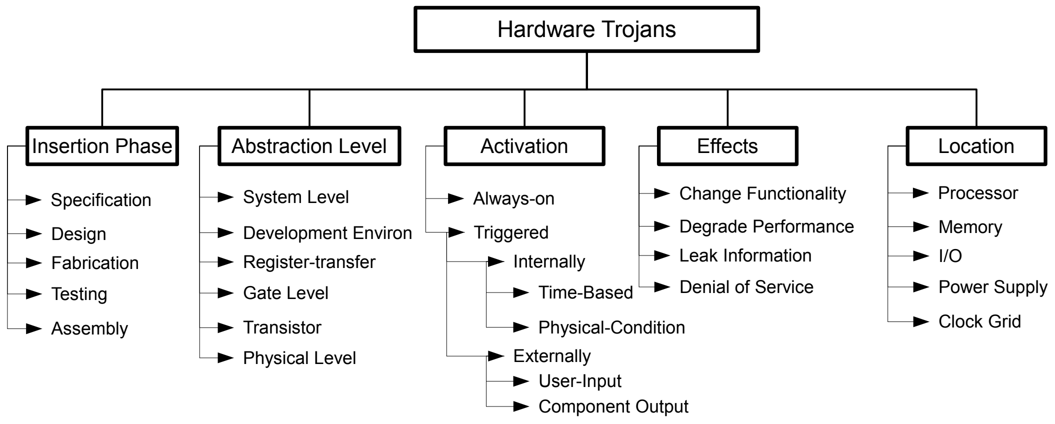

Figure 3-2: Hardware Trojan Taxonomy

According to this taxonomy, the aforementioned thinned wire scenario has the following attributes:

- **Insertion Phase:** Fabrication
- **Abstraction Level:** Physical Level
- **Activation Mechanism:** Always On
- **Effect:** Denial of Service
- **Location:** depends on implementation

While on the other hand, the Illinois Malicious Processor would have the following attributes:

- **Insertion Phase:** Design
- **Abstraction Level:** Gate Level
- **Activation Mechanism:** Triggered Externally via User Input
- **Effect:** Change of Functionality
- **Location:** I/O

While this taxonomy represents five attributes which characterize hardware Trojans, the three most important are the phase of insertion, their activation characteristics, and the Trojan effect, or Payload. Details regarding how an attack could be inserted during different phases of circuit production will be described in Chapter 4.

3.3.1 Activation Characteristics

The activation characteristics describe the vehicle through which the Trojan would be triggered to deploy its payload. Depending on how the Trojan is activated, difficulties with detection can change significantly.

3.3.1.1 Always-on

By far the simplest type of activation is the one that is always on, not requiring any sort of stimulus to begin undermining the function of the circuit. Generally this type of Trojan would have an unnoticeable payload, and continuously runs with no visible change. This type of Trojan can be difficult to identify due to the lack of switching-characteristics; some Trojan detection schemes rely on identifying a triggering behavior as a sign of a Trojan [33].

3.3.1.2 Triggered Externally

This type of Trojan represents the most commonly held image of a Trojan hiding a payload until given a specific signal. Such a Trojan usually watches inputs looking for a specific input or sequence of inputs that are very unlikely to occur during circuit verification or normal use, yet can be supplied by an attacker whenever desired. Upon receiving some triggering instruction, the Trojan finally allows its payload to affect the circuit, performing whatever malicious intent with which it was created.

3.3.1.3 Triggered Internally

This type of Trojan represents a Trojan that sits inert and is unaffected by inputs to the circuit. This lack of reliance upon external inputs makes it very difficult to detect, at the cost of decreasing the flexibility which which it can be triggered.

3.3.2 Effect/Payload

The payload of the Trojan embodies the purpose of the Trojan, and is the effect the Trojan has on the circuit. Some payloads might be undetectable, others can influence the circuit in significant ways.

3.3.2.1 Denial-of-Service

The denial-of-service (DOS) Trojan is one of the simplest, as this type of Trojan simply stops the circuit from working whenever it is triggered. Sometimes this type of Trojan can be attached to a fuse or something vital in the circuit and make this a permanent failure. In other situations, it would be useful as a way of preventing a device from working for a specific period. The example of the Syrian radars failing hints to the use of DOS Trojans.

3.3.3 Leak Information

Trojans that leak information can be invoked to bypass security in high security applications. For instance, a cryptographic circuit with a Trojan leaking information might constantly emit the secret key to its encryption. This information could be transmitted through any number of outputs, as a Trojan could leak information through anything from patterns of an LED blinking at high-frequencies, to electromagnetic fields generated by on-chip oscillators.

3.3.4 Degrade Performance

Trojans that degrade performance slowly corrupt the results of the circuit in question. While this could be used as a slow DOS Trojan, its primary benefit comes from surreptitiously corrupting the results and impairing the function of the circuit over time. This could be used over the long term to poison results, or as an act of slow deliberate sabotage. This degradation could appear in any number of ways, such as through slowly corrupted outputs, through changes in temperature or voltage, or even through changes in operating frequency.

3.3.5 Changing Functionality

The most general type of Trojan payload is that of a change in functionality. This is intentionally generic, as it represents a catch-all for Trojans that often perform circuit-specific functions. For instance, the Illinois Malicious Processor allowed attackers to completely take over the circuit and run malicious code. A cryptographic block might instead have logic for an additional secret key, unknown to those using it. Trojans that fall into this type can be among the most interesting, but also the most difficult to identify.

One point emphasized by this taxonomy is the sheer variety of possible Trojans that can be implemented. It would be impossible to identify a single countermeasure to eliminate all threats. The best that can be done is to focus on certain areas of this taxonomy, and remove partial threats. The first point at which to split Trojan detection methods is based upon the insertion phase of the Trojan. Among these, two phases are especially hard to completely perform in-house: Fabrication and Design.

Chapter 4

Production of Integrated Circuits

To identify possible threats to integrated circuits (ICs), it helps to define the steps in the production path The entire production of an IC can be broken down into four general stages. These stages are:

- Specification
- Design
- Fabrication
- Testing

Depending on the complexity of an integrated circuit, these individual stages can be comprised of many complicated steps. Furthermore, threats at each stage in production differ greatly, and require significantly different methods for mitigation. Some of these stages, such as fabrication, are often outsourced to external companies to save time and resources, and therefore represent a more significant threat. Other stages, such as specification, are almost entirely performed in-house, minimizing the likelihood of an external influence causing a malicious modification to a circuit.

4.1 Specification

The specification stage of production represents the point at which the requirements for the circuit are determined. In many cases, this represents decisions of protocols used, performance requirements, compatibility requirements, as well as overall intended function. In most cases, this stage can be performed entirely in-house, as there are no specific hardware or software resources necessary to perform this step effectively. The input to the design specification will be significantly simpler and easier to analyze than the input supplied in later stages. Furthermore, changes to the entire design are inexpensive at this point, and can be made with little difficulty.

Attacking circuit production at the specification stage is a difficult prospect. Even if external consultants successfully sabotage a design, hidden functionality could be easily changed in a subsequent update to the specifications. Furthermore, successful attacks would be hard to hide, as they would essentially represent an intentionally shoddy design. If these specifications were examined by a third party, then it is likely they would be discovered.

4.1.1 Protocols

One of the few possible places to attack a design at the specification stage would be at the point of deciding what protocols to use. If a protocol with an inherent security flaw is included in the design, then it could introduce a method for subverting the design in the future. This same attack would also work by implementing a low-security protocol in a situation where a high-security protocol is necessary. Another method might be to introduce vague specifications that appear to supply a level of security, but fail to deliver.

For instance, Koescher et al. [34] demonstrated how a low-level protocol, the Controller Area Network (CAN) protocol, was implemented over a car's internal network. This protocol was then secured on individual Electronic Control Units (ECUs) through challenge-response pairs in order to secure this insecure protocol. Unfortunately, this arrangement introduced an unbelievable amount of security flaws. As an example, despite the use of a supposedly secure encryption algorithm to generate these challenge-key pairs, the keys were only 16 bits, opening it to a brute-force attack. If a secure protocol was to be implemented in the first place, this would not have occurred. Unfortunately, access to this automotive system was shown to allow for malicious acts such as disabling brakes, lights, and instruments, and locking the doors (as well as disabling electric locks so the occupant could not unlock the doors electrically). It would be straightforward to implement code which selectively disables the brakes when the car moves at high speeds, and subsequently erases itself to hide the evidence of an attack.

4.2 Design

The design stage is the period in a device's production during which the device is fully mapped out to meet the required specifications. For these purposes, a complete digital representation of the final circuit is created. This is an incredibly important stage with regard to security, as there are a significant number of external influences to a circuit introduced during this stage. Few organizations can afford to completely create the design in-house, without relying on any external resources. Furthermore, the circuit at this stage can be convoluted, making it difficult to identify possible malicious modifications. This stage, in combination with the fabrication stage, represents the largest risk to the device.

With the large number of external influences on the design stage also comes a large possible variety of attacks. Furthermore, many of these external resources will often pass unchanged, or even unobserved, into the final design.

4.2.1 Third-Party Intellectual Property (IP)

The most significant vector to attacking a circuit during the design stage comes through the inclusion of third-party IP in a design. Most organizations cannot afford to build an optimized design from the ground up every time a common component is used, and therefore rely on IP vendors that supply design-blocks that perform the desired functionality. This then allows the organization to save money and time, avoiding the issue of creating the circuit from scratch. Designers will instead assemble licensed design-blocks to meet the requirements of the circuit, often treating the third-party IP as black boxes. These unknown designs can easily make their way unmodified into a final design, allowing for an effective vector for compromising a circuit.

For instance, suppose a designer was licensed a cryptographic circuit for use in a design. The cryptographic block's encryption could be easily undermined if it possessed an extra hidden key. While it would appear to function correctly under normal use, someone with knowledge of the hidden key could circumvent any security provided by the cryptographic block to the final circuit.

Another risk with third party IP is that there are a plethora of vendors supplying designs for every possible function, with very little oversight. Vendors come and go, often only possessing an online presence. It would not be difficult for someone to create a fake vendor persona and supply malicious design blocks at a below-market fee. It is simply not safe to trust the vendor to have clean designs, performing exactly as advertised. Compounding the problem is the reoccuring issue of stolen IP design blocks. There are a great deal of problems with vendors having IP stolen and resold by other vendors, or even just stolen by designers wanting to use the IP for free. Unfortunately, this has led to a culture of obfuscation and suspicion, making it difficult to obtain clean, unobfuscated code in order to identify possible attacks.

4.2.1.1 IP Watermarking

Another challenge with identifying attacks in third-party IP is the concept of IP watermarking. When licensing IP from a third-party vendor, that vendor will not necessarily be straightforward with the licensee when it comes to functionality within the design. In many cases, this is to protect the interests of the vendor, and prevent IP theft. A digital watermark is performed by embedding verifiable proof of ownership into a functional block [35, 36]. This proof can then be invoked at a later point in time to show whether a design was stolen or not. Unfortunately, the method through which the watermark acts is often functionally identical to that of a hardware attack leaking sensitive data. For instance, the study by Becker et al. [37] covers a method of watermarking a design such that a uniquely identifying string is broadcast through the power drawn during certain phases of activity. This data is transferred using a range of frequencies using spread-spectrum techniques to combat ambient noise. However, this uniquely identifying string could be replaced with sensitive information in the design such as a key or plaintext value. This change would convert this legitimate watermark into a full-blown insertion causing a data leak. Another similar study was conducted by Ziener et al. [38] which describes a method of watermarking for multiple IP cores in a single design, such as in a field programmable gate array (FPGA). This demonstrates an effective method of multiplexing these power watermarks so that each IP core can supply a different signature at the same time, without interfering with the watermarks of other cores.

4.2.2 Electronic Design Automation (EDA) Tools

Another example of an external resource used during the design phase is that of the tools used to create the design. If a backdoor is built into the tools used, then the design composed via those tools could be compromised. This is also a very difficult step to perform in-house, as developing design tools is an incredibly expensive process. If it were not, then companies such as Cadence and Synopsys would not be able to operate. However, unlike vendors for third-party IP, there are significantly fewer vendors providing tools, and there is more oversight with regards to tools used in designs.

4.2.3 Cell Library

Another external resource commonly relied upon during circuit design is that of an external cell library. While some organizations will have their own library to which they synthesize designs, many will need to rely on externally provided libraries, which might have internal faults. Despite this reliance, the cell library is not a very dangerous attack vector during circuit production. Due to the primitive nature of these cell libraries, it would be nearly impossible to introduce any sort of malicious attack beyond a simple degradation of performance. Furthermore, cell libraries are small enough that it can be possible for an organization to investigate one manually to verify minimum functionality.

4.2.4 Device and Component Models

Similar to how many organizations rely upon an external cell library, component models are commonly implemented in a design. Since these models represent low-level components such as transistors, resistors, and capacitors, this represents an abstraction level lower in the design than the cell library. Because of this low abstraction level, specific attacks would be very difficult to implement.

4.2.5 Hardware Description Languages

While it is unlikely to represent an attack vector, circuit designs also rely on externally developed hardware description languages (HDL) such as Verilog of VHDL. This is due primarily to the industry's development of electronic design automation (EDA) tools designed to work using these languages. While Verilog and VDHL are unlikely to inherently contain possible threats to a design, reliance upon an external language that is not as publicly tested could introduce errors or risks. An example of an external language that could be relied upon in a specific circumstance is the Symbolic Model Verifier (SMV) language provided by Carnegie Mellon University [39]. If an organization was to rely upon an external tool provided by an untrusted third party, then a possible attack could be hidden in the language used by the tool, rather than the tool itself.

4.3 Fabrication

The fabrication stage is the period in the production cycle where the chip is converted from a digital nonphysical design into an actual physical chip. This is a specialized process that has been developing since the mid-20th century, as fabrication plants compete in developing the best fabrication processes. This drive towards development is because the most advanced processes will create chips that are smaller, faster, and use less power than their competitors. More efficient processes can also reduce the number of errors and increase the yield of each lot. The steps in this stage require expensive equipment that is out of reach for all but the wealthiest organizations. A facility capable of modern fabrication processes costs several billion dollars to create [40], and requires continual upkeep and maintenance. Most circuit designers must rely upon an external third-party fabrication plant (fab) in order to have their design fabricated.

This reliance on external technologies is compounded by the difficulty in maintaining oversight of a design during this stage. Many of the largest external fabs are overseas (see Table 4-1), and plants can often be protective about details of the inner-workings of their processes, to protect trade secrets.

It is important to note that of the top 10 fab plants, only two are in the United States. It could be a serious risk in the supply chain if a contractor hired to create a product on a military contract was to make use of a chip fabricated overseas without recognizing the possibility of a chip being compromised.

Table 4-1: 2011 Top-10 Semiconductor Foundries [1]

Rank	Foundry	Location	Sales (USD)
1	TSMC	Taiwan	14,600M
2	UMC	Taiwan	3,760M
3	GlobalFoundries	U.S.	3,580M
4	Samsung	South Korea	1,975M
5	SMIC	China	1,315M
6	TowerJazz	Israel	610M
7	Vanguard	Taiwan	519M
8	Dongbu	South Korea	500M
9	IBM	U.S.	445M
10	MagnaChip	South Korea	350M

4.3.1 Masking

The first step in the fabrication process is the creation and implementation of the photolithographic masks used to fabricate the integrated circuits. Generally, the semiconductor fab plant is provided with a digital layout representing what materials are needed in what places, and at what levels. The masks are created at the fab based on this information.

Unfortunately, this stage represents an ideal opportunity for an adversary to introduce a hardware Trojan to the integrated circuit. Manipulations of the masks represent physical changes in the circuit. A creative attacker could make any desired changes to the layout prior to creating the masks, and there would be no way the designer could know about the change. This is the only stage of circuit production during which the attacker has complete control over every aspect of the final chip. Even if the designer could request and receive a copy of the masks, there is no guarantee that it would match the masks used for fabrication.

4.3.2 Fabrication

Declaring the next step “Fabrication” is an oversimplification, as the actual fabrication of integrated circuits is an ongoing series of process steps involving the masks [40]. This process involves multiple iterations of depositing polysilicon, dielectric or metal films, and growing oxide films, along with etching, and photolithography or impurity doping. This process is carefully managed, taking place in cleanrooms to reduce impurities. It would be very difficult for an attacker to make any changes at this point, as this is a delicate process with limited flexibility. Furthermore, any attacker in a position to intervene at this point would also be in a position to modify the masks, which would be a much easier way of making significantly large modifications to the design. At most, an attack at this stage would be performed through an introduction of impurities into the design to reduce reliability.

4.3.3 Wafer Probe

The result of the fabrication and masking steps are silicon wafers with multiple chips on each wafer. Fabrication plants use this opportunity to test each circuit with a series of test vectors in order to identify which circuits are functional and which circuits failed to fabricate correctly. At this point, the circuit is unchanged, so there is no opportunity to modify the chip. However, if this step is performed at an external fab, it would still be possible for an attacker to swap a wafer containing safe chips with a wafer containing Trojan chips. Another option would be to introduce several wafers containing Trojan chips into the entire batch to supplement a low yield, and salt the overall production with corrupted chips.

4.3.4 Dice & Package

After properly functioning chips are identified, the individual chips are cut out of the wafers, and connected to larger packages to allow for use. While this stage requires expensive equipment that might not be available for small organizations, it is possible to perform this step in-house. An adversary performing this step could put counterfeit chips inside the packages in place of the intended chips, or perform sabotage, such as attaching the chips in a subpar standard.

4.4 Testing

After the fabrication stage the individual packaged chips are subject to a large amount of testing in order to make sure that the designs work as intended. This step can be quite complicated, depending upon the complexity of the chip. This can require expensive testing equipment, and a significant investment of time to fully verify a circuit. While this step can be performed entirely in-house, outsourcing it to save costs introduces an opportunity for an attacker to replace chips with compromised ones. Generally the test vectors chosen will be completely trusted. The test sequences can be chosen in-house and can be supplied entirely from a trusted ATPG algorithm.

4.4.1 Test Equipment

While an organization can supply their own test vectors for a design, it is more difficult to design and supply their own testing equipment. As this equipment is entirely responsible for determining the validity of the tested chips, there is some leeway in using this equipment as a vehicle for attacking the chips. For instance, the testing equipment could arbitrarily supply high levels of power to circuits in order to cause damage and inhibit future performance. While this is a very small risk, it could mean the difference between utilizing inexpensive equipment from an unknown overseas company and expensive equipment from a larger more familiar company.

Chapter 5

Securing Production

Although these steps might introduce possible attack vectors, not all pose an actionable risk. It is crucial for a designer to determine the importance of securing the circuit, as well as what possibilities are available for compromising the circuit. For the majority of designs, many of these attack vectors can be ignored, because they bring about only a small risk and because of the difficulty of implementing an attack via such a vector. On the other hand, designs that require a level of trust require that certain steps be taken to reduce the threat provided by each of these steps.

5.1 Specification

In general, the threat of attack during the specification stage is low. Even if outside resources are implemented, complicated attacks are difficult to implement and much easier to identify than attacks inserted later in production. Furthermore, for a secure design, this stage could be performed in-house with little extra cost, eliminating any outside risks.

If use of an outside specification is deemed necessary, or the chip designers are provided specifications by an outside source, it is still possible to secure this stage. Several possible steps are available. For example:

1. Ensure secure information travels only over secure protocols
2. Avoid non-standard protocols
3. Avoid protocols with known problems
4. Use an outside source to vet the specifications

This is by no means an all-encompassing list, because there are many options that would increase the likelihood of a secure end-design. Furthermore, most of these options represent good design practices, so there is little lost in their implementation.

5.2 Design

Unlike the specification stage, the design stage is difficult to secure effectively. There are many parts that can come from external sources that make their way into the final design at this stage. However, the risk from these parts can be mitigated in most cases.

5.2.1 Third-Party IP

As third-party IP is supplied from an external source, there is no baseline with which to compare the IP to in order to identify differences. Instead, it becomes necessary to identify possibly suspicious behavior in a design. This means that the IP design needs to be thoroughly analyzed for possible malicious functionality.

Even though an absolute digital design is available, having a person manually search a suspect design for a possible attack is ineffective. One of the reasons for this is that there is no definite stopping point in a manual inspection. Having not found an attack does not absolve the design of threat, it simply means that no threat has yet been found at that point in time. Another reason is that most suspect designs will be clean and unmodified, so the vast majority of searches will yield negative results. Any such search of a design would be equivalent to looking for a needle in a haystack, where the needle most likely does not exist. In order to make such a search cost-effective, it is necessary to use a technique that performs this investigation automatically.

The most difficult part about automatically identifying an attack in a digital copy is that it is impossible to know what the attack will look like. It would be folly to assume that the attack only comes in one form or causes only one type of problem. In some cases it may be acceptable to only focus on attacks forming Trojans, which come with both a trigger and a payload. However in other cases, it is necessary to identify both bugs and malicious modifications together, as they have the same resulting effect on the circuit. This difficulty is compounded with the inclusion of IP watermarking in many designs; they will both appear as additional functionality hidden within a IP design block.

5.2.1.1 Recent Studies in Detection

Most studies that focus on identifying attacks in third-party IP focus on a specific type of attack. For instance, Wolff et al. [33] introduce a method of identifying hardware Trojans by searching out and isolating trigger/payload functionality in design blocks. More specifically, the technique focuses on the rare input-sequence that satisfies the trigger, and the change in functionality caused after this rare input-sequence occurs. However, this technique is limited to Trojans that trigger directly after a rare input sequence. This technique is ineffective in identifying malicious insertions that activate automatically or lack a standard trigger.

Another interesting study by Waksman and Sethumadhavan [41] focuses on isolating specific functional blocks and identifying when such a block begins to exhibit suspicious behavior. This is intended to be a method of verifying the trustworthiness of all blocks in a pipeline, even if they were designed in-house. This verification is implemented through a separate on-chip monitor-structure that constantly inspects the activities of the module in question in real-time. However, this technique is limited to pipeline modules and memory units, and only checks for triggered attacks that modify the output functionality of the module in question.

Unfortunately, there is a lack of investigation into methods for verifying the trustworthiness of an existing block. For this reason, a new method of identifying possible attacks in an IP block has been brought forth in this dissertation.

5.2.2 Tools

Fortunately, the tools used for circuit design can be mostly secured with little difficulty. The large vendors such as Cadence and Synopsys can mostly be trusted to supply tools without hidden functionality, and the smaller vendors supplying tools can be investigated if suspicious, or avoided if necessary. Unlike the situation with third-party IP, there is a limited supply of tools available for use in circuit design. This limited supply has therefore been much more thoroughly tested and evaluated by others using these resources. While this step poses a risk to the unaware designer relying on untrusted tools, a cautious designer could avoid most threats.

5.2.3 Cell Libraries and Models

While external cell libraries and models are resources provided from sources outside the organization, they are significantly easier to analyze for corruption. Unlike third-party IP, the functions of these cells and models are clearly stated, with no hidden functionality buried within. The actual cells and models can be manually inspected to identify anything suspicious. Since cells and models are often used successively between circuits, this analysis would be a one-time fixed cost that could be spread out among all designs.

5.2.4 Hardware Description Languages

While it would be difficult to develop an in-house language for use with circuit designs, the most commonly used outside languages are unlikely to have risk. VHDL, for instance, was developed by the U.S. Department of Defense, and eventually standardized as IEEE 1076 at the behest of the U.S. Air Force in 1983 [42]. Similarly, the Verilog language has been standardized as IEEE 1364. Based on the history of these open standards languages, it is unlikely that any inherent security flaws are present.

As long as the designers rely upon open standardized languages, the overall risk from this external source is minimal. In fact, this is one of the main purposes of forming open standards, as it allows for universal verification of the integrity of these languages.

5.3 Fabrication

When it comes to identifying attacks in a circuit, the point of modification is critical in determining the techniques used to identify a possible modification. For a circuit attacked during one of the fabrication steps, the only way to identify the attack is with a thorough investigation of the resulting physical chip. However, this is not an easy process, as there are confounding factors that make it difficult to identify an attack on a physical chip including: (1) the small feature size and (2) process variation.

5.3.1 Confounding Factors

The first of these factors is simply the feature size of the circuit, since modern fabrication technologies are measured in nanometers, and proceeding to get smaller every year. While there are many imaging techniques available, these are often not an effective method of validation. For instance, physical imaging techniques such as cross-sectioning and layer removal/deprocessing require the destruction of the chip. Thermal imaging techniques, such as infrared thermography, and optical techniques, such as scanning optical microscopy, can be effective in mapping out a chip, but they still often rely on individually activating sections of a chip. Additional techniques such as those relying on electron beams can be effective at mapping out a circuit, but are also expensive and can cause adverse effects due to depositing charged particles in the circuits [43]. The most substantial problem with all of these techniques is that they are too expensive. Even if imaging an individual chip does not cost much time or money, it is simply not feasible for all of the chips being produced to be similarly imaged and analyzed for modifications. This is even more significant of a monetary cost when dealing with imaging techniques that risk destroying the chip. Any effective method of identifying modifications in such a design must be both nondestructive and inexpensive.

5.3.1.1 Process Variation

The second confounding factor, and the most important, is a phenomenon known as "Process Variation". This has many names, including process variation, process variability, and variation within fabrication processes. These variations are random fluctuations in the design caused by the technology used in manufacturing the chip.

Individual variation between lots, dies, and even individual chips within a die can have a significant impact on circuit functionality [44]. Furthermore, this variation increases with each subsequent generation of integrated circuit (IC) manufacturing. In 2005 for state-of-the-art fabrication plants, leakage current alone was estimated to vary by 5 to 10 times, causing an almost 50 percent variation in total power [45]. This impact has likely increased over the past 8 years. Considering how a chip's power consumption can vary by up to 50%

from process variation, it would not be strange for the impact from hardware Trojans to be masked by such fluctuations.

5.3.2 Side-Channel Measurements

The most promising methods for detection currently use nonstandard side-channel measurements, such as the power drawn by the circuit, or the timing for individual outputs. There are even studies leveraging changes in a magnetic field, or individual light emissions [46, 47]. Generally these side-channel measurements are based on techniques in cryptography known as side-channel attacks, where cryptosystems are broken using these measurements. In addition to being non-destructive, such measurements are significantly less expensive than imaging techniques. However, since these measurements are susceptible to process variations, it is important to develop a technique that is able to garner reliable results regardless of these fluctuations. In most cases, reliable results can only be obtained if the tester has access to a "golden copy", or a known trusted copy of the chip. However, obtaining such a chip is often significantly more expensive.

5.3.3 Current/Power analysis

Current or power side-channel measurements are presently the most effective techniques available for identifying Trojans in fabricated circuits. The basic concept behind these techniques involve measuring the current drawn by the circuit over various inputs, and comparing that current with the measurements taken from a golden copy.

One of the first studies implementing this type of technique is by Agrawal et al. [48]. The study focuses on the use of integrated circuit (IC) fingerprints, which are generated through testing and validation of a golden copy made at a trusted fabrication plant. These fingerprints are used with standard signal-processing techniques to reduce the effect of process variation, and identify the presence of Trojans in synthesised RSA circuits. The process variation ranges which they tested were from 2% maximum variation up to 7.5% maximum variation, and through which they could identify Trojans that were at least 0.01% of the size of the original design, with a 2% false positive rate.

Another good baseline study for this type of technique is by Wang et al. [49] where the transient current from a suspect IC is compared directly with the current drawn from a safe golden copy. To overcome process variations, the difference in the two currents is integrated over time and compared with variation detected on the golden chip. These tests were run on the ISCAS benchmark s38417 circuit with two types of Trojan triggers: counters and comparators. The tests showed that Trojans with triggers of similar size can be detected through minor process variations when a golden copy is available. The range of variations covered include a threshold voltage change of up to 5%, a change in channel length of up to 2%, and a change in oxide thickness of up to 1% for inter-die variation. However, this amount of variation is insufficient to represent the variations present in modern fabrication processes, which can be up to an order of magnitude larger.

A study by Rad et al. [50] shows similar results to the Wang study but introduces another interesting method of testing: reducing the size of the tested circuit by focusing on individual powered sections of the chip. Though this study did not focus on the issue of process variation, this technique is useful in combating such variation. Focusing on individual powered sections of a chip effectively reduces the area of the chip, therefore reducing the amount of process variation adding noise to side-channel measurements. This optimization can be combined with other methods involving process variation to improve efficiency.

A similar method of segmenting a circuit to reduce the effect of process variation is tested by Banga and Hsaio [51, 52]. In these studies, several ISCAS circuits are segmented through careful selection of input vectors, with the purpose of identifying regions suspected of containing possible Trojan circuitry. While these studies do not directly address the problem of process variation, the techniques developed in these studies would be most useful in combination with other techniques for the purpose of combating high amounts of process variations. As with the Rad et al. [50] study, reducing the effective area of the chip during tests reduces the effect of process variations on side-channel measurements.

Other interesting methods include another study by Banga and Hsaio [53] where the supply voltage and ground are inverted as a method of identifying unlikely triggers. The idea behind this type of structure is to pick an input with as many active sections of the chip as possible and then invert the power to the chip. After inversion, the active sections and inactive sections will swap. Since Trojan triggers are small inactive sections that do not change on most inputs, they can be more easily activated and then measured.

There are many more studies implementing current analysis as a side-channel measurement for detecting Trojans (e.g., [54, 48, 55, 56]). However, all of these methods either require a golden copy to calibrate the measurements or do not address the issue of process variation.

5.3.4 Timing/Delay Analysis

The other main type of side-channel measurement involves high-speed timed measurement of responses from a change in an input. The purpose of such a measurement is to identify the gate-delay from additional combinational logic introduced by malicious insertions (typically Trojans). Additionally, many countermeasures for attacks involving timing analysis include preventative structures. Such structures are intended to make identification of an insertion an easier task. However, process variation generally has a greater impact on timing measurements, rendering most of these techniques ineffective in the face of high levels of variation.

One of the first circuit-hardening techniques was introduced by Li and Lach [57]. This study introduced the use of shadow registers [58] as a method of identifying the possible presence of malicious circuitry. Normally such registers would be used as a method to verify the logical integrity of the circuit or to debug abnormal behavior. However, if these registers are connected to a tunable clock input, then they can be used to accurately measure the response time of specific inputs to individual registers. Since any malicious insertion requires the addition of extra gates, any delay from these gates would be measured by the shadow registers. Unfortunately, large process variation ends up making this technique ineffective as a standalone solution, as shown in a subsequent study by Rai and Lach [59]. However, it still remains a useful tool when combined with some of the other techniques described here.

Of course, attack detection using timing and delay analysis does not necessarily require predefined structures. Jin and Makris [60] tested a technique where a fingerprint generated for a DES circuit synthesized to an FPGA can be used to detect Trojans larger than 0.13% of the size of the circuit, with up to 7.5% variation between cells. However, these tests were only effective on Trojans with an explicit payload that alters the functional logic, and affects values on output pins. Malicious modifications with an implicit payload were undetected.

Unfortunately, due to the strong effects of process variation on delay-classification [61] of a circuit, it is generally less effective to determine the existence of attacks through timing measurements, than through power measurements.

5.3.5 Circuit Trust through Hardening

Another method of mitigating possible attacks on a fabricated design is through pre-emptive hardening techniques. These techniques can either try to make a circuit more difficult to attack (obfuscation), or they can make intrusions more easy to detect. These techniques can usually be used in combination with other side-channel techniques involving current or timing analysis.

For example, Koushanfar and Mirhoseini [62] describe a formal framework for designing integrated circuits such that side-channel measurements can be taken from individual modalities easily. Additionally, implementation of the framework reduces the amount of noise at each measurement, combating the effect of process variations on such results. The study uses its own side-channel analysis involving current drawn from the circuit. However, their framework could also be used in combination with techniques proposed by other researchers to more fully cover possible attack vectors.

Another study by Chakraborty et al. [63] describes an IC design methodology that facilitates detecting trojans inserted during fabrication. This methodology makes identification of Trojans easier through logic testing and also obfuscates the design to make inserting a trojan significantly more difficult. This testability comes from breaking the circuit into several functional modules and then switching into a “transparent mode” where the circuit simulates the occurrence of rare inputs that might act as a trigger for an attack. The obfuscation caused by this methodology is a useful tool for increasing the cost of attacking the design. If the attackers cannot identify what parts of the design block have what functionality, then they cannot easily insert malicious circuitry to modify that functionality.

5.3.6 Circuit Trust through Obfuscation

While it is not always possible to prevent modifications to a design, or even to identify the changes to a design, one option for preventing the inclusion of hardware Trojans is to make it difficult for the attackers to identify the function of the design. An example of this would be a side-effect of the twin-fab production concept [64, 65, 66]. This method was initially conceived as a way of countering design theft during the fabrication step, but would also make attacking the design through modifications or insertions very difficult, if not impossible. The main concept behind this method is to rely upon the external fab for only the lowest layers of the chips. This allows a designer to only send the bottom-half of the layout to the fabrication plants, hiding the functionality of the chip. The half-completed wafer could then be forwarded to a trusted fab plant with a larger feature size to attach the higher metal layers that do not require a small feature size. Since an attacker would not possess the full design, they would be limited in the changes that could be made, and would also be forced to reverse-engineer the design to determine what to attack.

5.4 Testing

In general the testing phase does not represent much of a threat. While it would be risky for the design to be tested by someone outside of the organization, this step can be easily kept in-house. While there is a risk posed by making use of externally supplied testing machines, this could be avoided by simply using machines from a reputable company, purchased through a reputable supplier.

5.5 Summary

Full assurance that a design is secure requires preventing or detecting malicious influences at every stage of the circuit production process. While the specification and testing stages can be attacked by a resourceful attacker, preventing modification at these stages does not pose a difficult task, as both of these stages can be performed in-house limiting external influences. Unfortunately, the design and fabrication stages of circuit production rely significantly on external resources; third-party IP is a necessary inclusion of many chips during the design stage, and the fabrication stage heavily relies upon out-of-house foundries which are often overseas and untrusted. Securing these two stages of the design process must take a priorit in further research.

Chapter 6

Detection of Hardware Trojans in Third-Party Intellectual Property using Untrusted Modules

Following the order of a design flow, this chapter will focus on security pre-fabrication. To identify hardware Trojans hidden within 3rd party design blocks, this research applies a method that is similar to fault tolerance testing using design diversity [67, 68]. Dubbed Design Comparison, this method contrasts two designs of dubious nature providing similar, but not identical, functionality to validate each other. This process is effective despite the fact that neither of the two designs are trustworthy. This effectiveness comes from the Trojan's inherent secretive nature. Trojans can fundamentally be described with two parts from the taxonomy in Chapter 3: (1) their *payload*, which is the effect they cause, and (2) their *trigger*, or the inputs and condition that cause the Trojan to deliver the payload [69]. By design, triggers are intended to avoid activation during normal testing and usage so they remain undetected. Based on this approach, if an individual Trojan is unlikely to be triggered by accident, then it is even less likely for two Trojans to trigger in the exact same way, with the exact same payload. Identifying any inherent logical difference between the two designs will therefore identify any Trojan functionality.

To compare the two designs effectively, they first must be prepared. Standard verification techniques are not intended to compare two structurally different circuit designs, especially when they often have different inputs, outputs, and use. The two processing steps are to: (1) wrap the two designs to match their functionality, and (2) unroll all of the internal state components, allowing for the design to be expressed entirely in terms of outputs and inputs over time. This process then allows for each output from the two circuits to be compared with a Boolean satisfiability solver. This enables the identification of the set of possible inputs that could cause the two circuits to provide a different output.

Design Comparison was tested over a variety of standard benchmark circuits of a range of sizes, and was overall found to be very effective detecting malicious modifications to third-party IP. The testing time for the analysis was generally equal-to or less-than the time it took to run the circuit through standard circuit Electronic Design Automation (EDA) tools for synthesis.

The key contributions from this work are:

- Development of a technique for identifying malicious modifications in third-party IP using two unverified modules.
- The testing and validation of this technique over a series of benchmark Trojans.

6.1 Design Comparison

The primary concept behind Design Comparison is to logically compare two untrusted designs with similar functionality, identifying wherever their outputs differ [70]. As Trojan triggers are designed to have extremely low odds [71] to prevent accidental triggering, the odds are even less that two Trojans would have the same Trigger and the same payload. With many vendors available from which designs may be selected, finding two similar designs for comparison is simplified. In the rare occurrence that a logical block has unique functionality that needs to be verified, testers could write a top-level Verilog design to instantiate the IP that focuses purely on functionality (instead of area or timing); this top-level design would be sufficient for comparison.

There are three main steps to the comparison process: (1) Wrapping the two designs, (2) Unrolling the state logic within each design, and (3) Comparing the two circuits according to Boolean satisfiability (SAT). The detection scope of design comparison must also be considered.

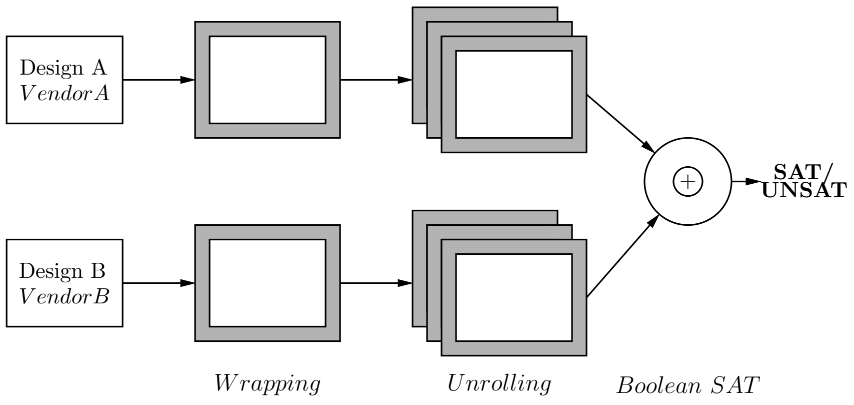

Figure 6-1: Design Comparison Overview

6.1.1 Wrapping

Even if two designs have the same essential functionality, their actual implementation of this functionality can be vastly different. The blocks might use different instruction codes, or have different ways of interpreting inputs and outputs. There are any number of minute differences that could exist between two blocks that have effectively the same functionality. The point of this step is to create a wrapper around each of these blocks to match their exact functionality. As this is purely logical, and ignores all optimization, the wrapper does not need to be well-timed or optimized for area. Furthermore, if a design is going to be included in a circuit, then much of this work would already be necessary to ensure each design works correctly within the circuit. However, there are a few challenges that must be addressed.

6.1.1.1 Don't-Care Inputs and Outputs

The comparison between the two circuits will find *every* difference, which will unfortunately also include differences as seen by don't-care situations. Take the example of an ISCAS89 s208.1 frequency divider. Given a set of inputs and clock pulses, thic design divides the frequency of the pulses based on the inputs. However, as this circuit is frequency-based, the output only needs to have the correct frequency. If a similar circuit was to produce an output that was 180° out of phase with the s208.1, then the outputs would technically be different, yet both correct. This type of difference needs to be rectified during the wrapping stage, or there will be false positives when the comparison is performed.

6.1.1.2 Different Capabilities Between Designs

Another problem that will arise during the wrapping step is that the two designs possess a slightly different functionality. For instance, one design might be strictly an Arithmetic Logic Unit (ALU), while the other design is a combination ALU and function generator. In this case, there exists known functionality in one design that is not present in the other. At this point, there are two possible ways to continue the wrapping.

- Use the design with the extra functionality (the ALU plus function generator) to verify the design with less functionality. This can be performed by simply ignoring the extra functionality in the wrapper, which effectively removes it from the design.
- Use the design with less functionality as well as another smaller design containing that missing functionality, to verify the larger design. With the ALU example, this would entail finding or creating a simple function generator that can be used to validate the extra functionality of the larger design.

Additionally, there is always the option of finding another design to verify. One of the advantages of working with third-party IP is that there are many alternatives, and if one design is insufficient for current needs, then another one would likely work. The first choice is also useful in case of difficulty in finding similar designs to compare. In this situation, a more complicated design can be wrapped in such a way that it only performs a small subset of its intended function, matching that of the design to verify.

6.1.1.3 Lack of Comparable Designs

While it is unlikely, it will always remain a possibility that only one vendor will sell the block that is necessary to complete a design. Depending on the complexity of this block, it is possible to create a similar block that performs identically for the sole purpose of comparing the two designs. As this new design has no requirements with respect to timing, area, or efficiency, it would be significantly easier to create than an actual design for final implementation. However, this still represents a significant investment of time.

6.1.2 Unrolling

After the designs have been wrapped, they are still implemented differently. Even though they now effectively have the same number of inputs and outputs, they likely have different structures, state components, and internal logic. To effectively compare the two, it is necessary to remove all internal state logic, and rewrite the outputs as Boolean expressions of past inputs. Consider the circuit in Fig. 6-2 as an example. This circuit contains two

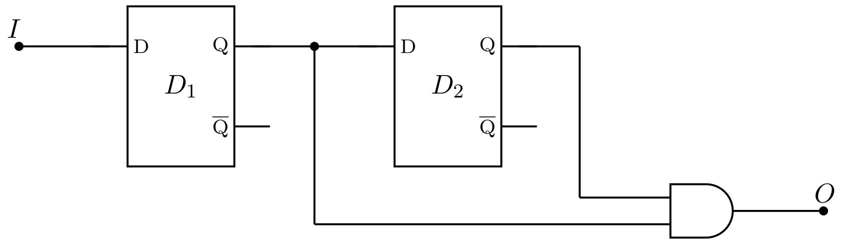

Figure 6-2: 2-bit sequence detector (for '11')

state-bits, as well as one input and one output. We can express the logic progression during a single clock cycle as a set of Boolean expressions (Eq. 1).

$$D_1 \leftarrow I, \quad D_2 \leftarrow D_1, \quad O \leftarrow D_1 \land D_2 \tag{1}$$

By stepping backwards from the output and decrementing a counter on the input, the flip-flops can be gradually replaced with inputs over a period of time.

$$O \leftarrow D_{1[t-0]} \land D_{2[t-0]} \tag{2}$$

$$O \leftarrow I_{[t-1]} \land D_{1[t-1]} \tag{3}$$

$$O \leftarrow I_{[t-1]} \land I_{[t-2]} \tag{4}$$

As seen in equation 4, the output is expressed entirely in terms of inputs, with no dependance upon internal state logic. However, not all circuits are this simple, and many possess circular logic that will never fully be removed while unrolling. To ilustrate this issue, consider the example circuit shown in Fig. 6-3. This circuit represents a 2-bit multiple-input signal register (MISR).

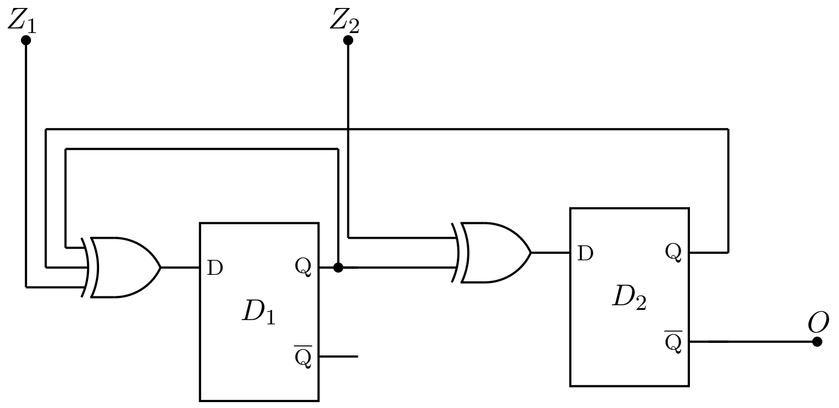

Figure 6-3: 2-bit Multiple Input Signal Register (MISR)

Unlike the sequence detector, the MISR has inherent cyclical logic, where the value for a flip-flop is based on its own value at some previous clock cycle. This is more presently visible if we unroll it several cycles.

$$D_1 \leftarrow D_1 \oplus D_2 \oplus Z_1, \quad D_2 \leftarrow D_1 \oplus Z_2, \quad O \leftarrow \bar{D}_2 \tag{5}$$

$$\bar{O} \leftarrow D_{1[-1]} \oplus Z_{2[-1]} \tag{6}$$

$$\bar{O} \leftarrow D_{1[t-2]} \oplus D_{2[t-2]} \oplus Z_{1[t-2]} \oplus Z_{2[t-1]} \tag{7}$$

$$\bar{O} \leftarrow D_{2[t-3]} \oplus Z_{1[t-3]} \oplus Z_{2[t-3]} \oplus Z_{1[t-2]} \oplus Z_{2[t-1]} \tag{8}$$

When unrolling a circuit with cyclic logic like that shown here, the process can be performed by simply unrolling to a specified *maximum depth*. This depth represents a window within which all possible combinations of inputs will be verified. Any state values beyond this maximum depth are arbitrarily set to 0. In the end, the goal is to ensure that all logic is dependent only upon the inputs and outputs, thus eliminating the state.

Another way to visualize this type of unrolling is to create N copies of a circuit (where N is the unroll depth) and to remove all state-logic from all copies (see Fig. 6-4). In exchange, anywhere a value is pulled from a state component (i.e., the Q port on a D flip-flop), the value is instead taken from the stored value value on the next circuit down the line (i.e., the D port on a D flip-flop). On the last circuit, state logic is given an arbitrary value, and the chain ends.

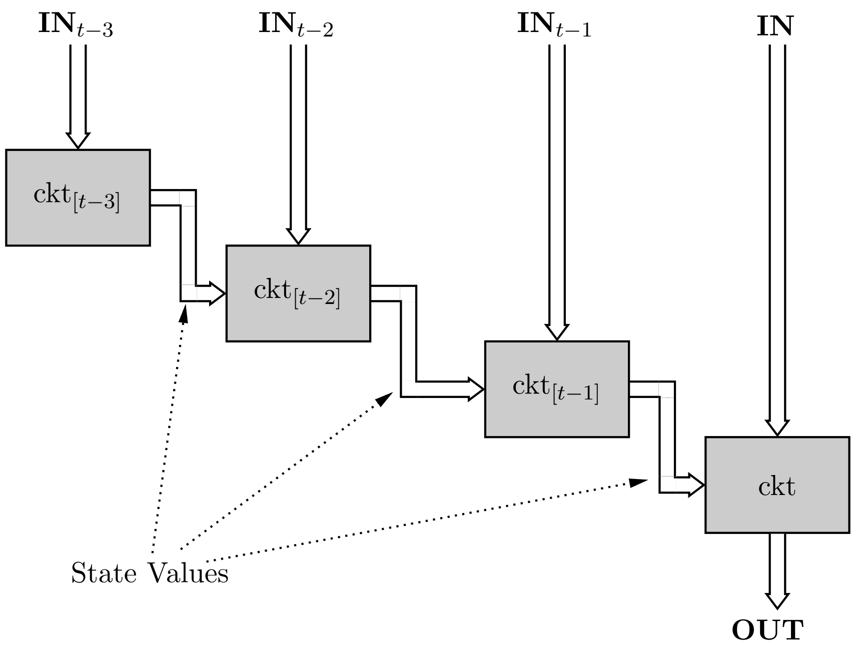

Figure 6-4: Unrolling through circuit duplication

The goal of the unroll depth is to provide a testable window which can be used to provide an estimate that is accurate *enough*. Testing over a window of depth 20 proves that there is no possible combination of 20 inputs that trigger Trojan logic. Depending on the level of accuracy and the computational power available, the window can be selected to meet necessary needs.

6.1.3 Boolean Satisfiability

The final part of testing is to run Boolean satisfiability tests between the outputs of the test circuits for each input. SAT can be used effectively to identify logical differences in Boolean expressions [72]. By solving for a situation where the output on each circuit provides different values (either through a negative Boolean satisfiability check, or by XOR-ing the two outputs) the solver identifies the exact sequence of inputs that provides the difference in outputs. This difference can then be examined to (1)) determine which circuit is correct, and (2) exactly what the circumstances were that might have caused such a difference. If the difference is considered to be a false positive, then the wrapper can be modified. By changing the wrappers to arbitrarily fix the difference seen over this series of inputs, the tests can be rerun to search for further differences.

6.1.4 Detection Scope for Design Comparison

One of the key limitations to this technique is that it focuses entirely on logical Trojans. If a Trojan has a payload that has no impact on outputs, then this technique is ineffective. However, Trojans with no impact on outputs are significantly easier to detect when investigating suspicious circuitry. This difference is because this circuitry can appear superfluous and inefficient, often containing special structures such as ring-oscillators that arbitrarily heat the circuit, or manipulate power consumption. Design Comparison is effective when paired with a detection script tasked with identifying strange component structures.

6.2 Evaluating Trustworthiness with Design Comparison

One of the most important evaluations of a technique like Design Comparison is the determination of what causes mistakes, namely false positives and false negatives. This section describes how both cases can result from Design Comparison.

6.2.1 False Positives

Even though the Design Comparison process might flag something as suspicious, there are several situations where it is not in fact a Trojan. These include don't-care situations from the wrapper, design errors in the module, and non-Boolean logic structures that raise suspicion.

- Failed Wrapper - This condition happens when the wrapper fails to account for various differences between the designs, such as don't-cares, differing input requirements, or other similar situations. If this is the case, then the only options are (1) to go back and add these specific situations to the wrapper or (2) to specifically edit the SAT equation to arbitrarily ignore such situations. With sufficient preparation on the wrapper, this condition should not occur.

- Errors in the designs - This technique will flag *any* logical difference between the two designs. If a design has an error or bug, then this will identify it and flag it as a possible Trojan.

- Suspicious circuitry - Any ring oscillator or self-driving loop that does not have a state-component to delay the loop will flag an error during unrolling. This condition also occurs when there are multiple inputs driving a signal or other nonstandard design techniques, such as using multiple undefined clock-signals.

Fortunately, none of the false-positives pose a significant problem for using Design Comparison. Don't-cares can be handled in the wrapper without significant effort, and design-errors must be identified anyways, as they can be just as dangerous as Trojans in a critical circuit. Ring-oscillators and other types of circular logic are quite often used in Trojans as one of the critical structures that deliver a non-standard payload such as power manipulation. If structures such as ring oscillators are identified in a design, then their use should be examined very closely, because they are uncommon when following standard design practices.

It is important at this point to once again mention the possibility of a vendor implanting IP watermarks into their offered designs. When the design comparison process identifies suspicious circuitry, it is possible for this circuitry to be a part of a digital watermark, making the flag a false positive (Section 6.2.A). However, it is important to examine such a positive carefully because of the similarity between such watermarks and Trojans.

6.2.2 False Negatives

Overall there are fewer situations that lead to a false negative result than those that generate a false positive. Generally these situations are limited to either a delayed payload, or a non-logical Trojan.

- Delayed Payloads - The most straightforward way to circumvent Design Comparison is to implement an arbitrarily long delay between trigger and payload, stretching wider than the test-window.

- Non-Logical Trojan - Again, Design Comparison does not try to identify Trojans with payloads that do not affect the outputs. Such Trojans will not raise flags with this technique, unless it is noticed during the unrolling step.

The key worry with a false negative is that a Trojan has too long of an implementation time to fit within the window formed by the unrolling depth. However, such a Trojan requires a large amount of circuitry, effectively requiring a large counter specifically designed to delay the payload. This scenario makes the Trojan easier to detect by techniques focused on identifying suspicious circuitry, such as those presented in [73].

6.3 Experimental Methodology

As a process, Design Comparison can be implemented in multiple ways, with a large variety of different tool options. To show a proof-of-concept and to test the effectiveness, the Cadence suite of tools (RTL Compiler and SMV) [74], AIGER [75], and relsat [76] were used for this work. Regardless of the tools used, the overall process remains the same.

All software and tests were run on a Dell PowerEdge T105 with an Intel(R) Xeon(R) 2.8 GHz CPU with 16 GB of RAM. While four cores were available, all tests were run on a single dedicated core with no parallelization. The following describes the methodology for each stage in the process.

6.3.1 Wrapping

The only manual step in the comparison process is the wrapping step, wherein the two circuits are encapsulated in external Verilog files that matches the inputs and outputs of the two circuits. This process involved:

- Forcing enable pins to 1 or 0
- Disabling scan chains
- Ignoring debug outputs not shared between designs

There were also occasional bugs or syntactic errors that were corrected at this point. It is important to note that when implementing a wrapper, only the wrapped circuit is tested.

If key logic ignored by the wrapper needs to be invoked in the final design, then it needs to be included in the wrapper. For this reason, it is both prudent and simpler to search for similar designs to test.

6.3.2 Unrolling Tools

The first key step in the unrolling process is to convert both test circuits to the same logical form. In some circumstances, the designs were written in functional Verilog, while in other situations, designs have been synthesized to a unit cell library, expressing everything in terms of simple and complex logic gates. This step is performed by reading the designs into Cadence RTL-Compiler, elaborating the design, and writing their output as a Boolean equation (with the "-equation" flag).

The second step in this process is to convert everything into a Symbolic Model Verification (SMV) specification language (specifically Cadence SMV [77]), which prepares the circuits for analysis. This is facilitated in part by some of the Cadence SMV tools, which can convert limited Verilog descriptions into SMV. As the SMV specification language and Verilog have many similarities in structure and notation, this conversion is straightforward.

The third part of the process is a mock unrolling, where circuits are unrolled ignoring the output. The purpose of this step is to identify strange behavior within the circuit that might either represent a Trojan or interfere with Boolean analysis. Furthermore, this greatly optimizes test time by identifying problems without requiring a full unroll and comparison process.

The fourth and final step is the actual unrolling process. To optimize this step, each circuit is duplicated N times (where N is the unroll depth), duplicating inputs and wires as well. These duplicates, representing past states, are then connected together to remove all flip-flops and other state-components. The initial value of all state components representing time beyond that window are set to 0.

6.3.3 Boolean Satisfiability

After the circuits are unrolled, all of the outputs need to be verified individually to identify differences. This can be accomplished by combining the two unrolled circuits into a single circuit sharing inputs, and with the outputs-to-test connected to each other via an exclusive-or (XOR) logic gate. If a logical situation occurs that causes the two outputs to have different values, then the output of the XOR will change to 1.

Although SMV could technically perform this test on its own, it is in an unoptimized form that can be greatly improved. One of the key reasons for converting the circuits to the SMV format is its interoperability with model verification tools, including the AIGER toolset. In this case, the AIGER utilities are used to convert the SMV-expressed problem in terms of and-invert graphs, and subsequently converted to the conjunctive normal form (CNF) in order to be fed directly into the SAT solver of choice, namely relsat. This process was subsequently repeated for each output, halting if a difference is identified. Unfortunately, this process must be repeated for each and every output, so circuits with a significantly large number of outputs require a long time to test if every single output needs to be tested.

6.4 Testing and Results

The Design Comparison technique was implemented over a variety of Trojans within several standard benchmark circuits. The benchmark circuits are from several sources, including OpenCores [78] and the International Symposium on Circuits & Systems (ISCAS), while the Trojan benchmarks are supplied from the Trust-HUB repository [79]. The size of these Trojans vary greatly, from a small RS232 Universal Asynchronous Receiver Transmitter (UART) to the large Wishbone Conmax Interconnect Matrix IP core from OpenCores. To demonstrate the full range of benchmark sizes, they were synthesized to the Oklahoma State University (OSU) 45-nm Standard Cell Library [80] and resulting sizes are shown in Table 6-1.

For the most part, the size of the circuit directly relates to the cost in time that it takes to test each circuit. The generic cost in time for testing at an unroll depth of 10 is shown in Table 6-2. The time it takes to run RTL-compiler is also listed because it provides an excellent comparison for the cost of unrolling and testing satisfiability in real time.

One of the most interesting results here is the time it took to run the satisfiability tests on

Table 6-1: Size of base benchmark circuits when synthesized to the OSU 45-nm Standard Cell Library using Cadence RTL Compiler

Benchmark	Size(um^2)
RS232 UART	1,269
PIC16F84	8,848
s15850	15,654
s35932	45,112
s38417	43,413
s38584	48,408
Wishbone-Conmax	126,986

Table 6-2: Cost in time to run test over different benchmark circuits (seconds)

Benchmark	Trojans	RTL	Unroll	SAT
RS232 UART	T200	2.33	1.86	0.0044
PIC16F84	All	1.26	-	-
s15850	All	22.41	23.36	0.037
s35932	T200,300	129.38	76.36	0.0053
s38417	T100	115.9	71.12	0.093
s38417	T200,300	-	-	4.12
s38584	T100	130.32	71.42	159.54
s38584	T200,300	-	-	41.22
Wishbone-Conmax	All	563.1	177.0	6.14

the s38584 and the s38417, because the test time varied significantly depending upon which Trojan was tested. Despite the Trojan circuits being extremely similar, the difference was more than an order of magnitude. This is likely due to the way the satisfiability problem was approached by the solver, resulting in an inefficient solution under certain circumstances. By running these tests under a different satisfiability algorithm, the results would likely change as well. The test results for each benchmark are broken into the following subsections.

6.4.1 RS-232 Universal Asynchronous Receiver/Transmitter

The first set of tests were run on an RS232 Universal Asynchronous Receiver/Transmitter (UART). This is a relatively small circuit, possessing only 12 input pins, and 11 output pins. There were 10 Trojan implementations for this circuit, labeled T100 through T901, representing 10 implementations of 9 different Trojans (with two versions of the Trojan T900, listed as T900 and T901). All of these Trojans were detected quite easily and quickly, as T100, T200, and T400-T901 were all identified during the unrolling process due to suspicious logic.

The final Trojan T300 was identified upon comparison to a different version of an RS232 UART using the design comparison technique so as to identify the Trojan. The two versions were sufficiently different, so that when each design was synthesized to the OSU 45-nm cell library, the Trojan circuit had a footprint of 1989.8 um^2, while the other took 1269.9 um^2. The Trojan was quickly identified in the malicious circuit with an unroll depth of 10 cycles.

To further explore the efficiency of the process, the unroll depth was arbitrarily extended to greater lengths, despite 10 cycles being sufficient for detection. The results in Fig. 6-5 show that the time taken for unrolling is consistently linear as expected. The most interesting results occur during satisfiability checking, as the results have a large amount of variation, yet still show a rough linear increase in test time. Plotting the graph on a logarithmic axis (Fig. 6-6) makes this easier to see. These results were calculated as an average of tests at each unroll depth, with very little measured difference in time taken between each test at each depth. This means that the tests at depth 240 and 250 were consistently taking approximately 340 seconds to complete, while the tests at 260 and 270 were consistently taking 30 seconds to complete. These differences in testing times are most

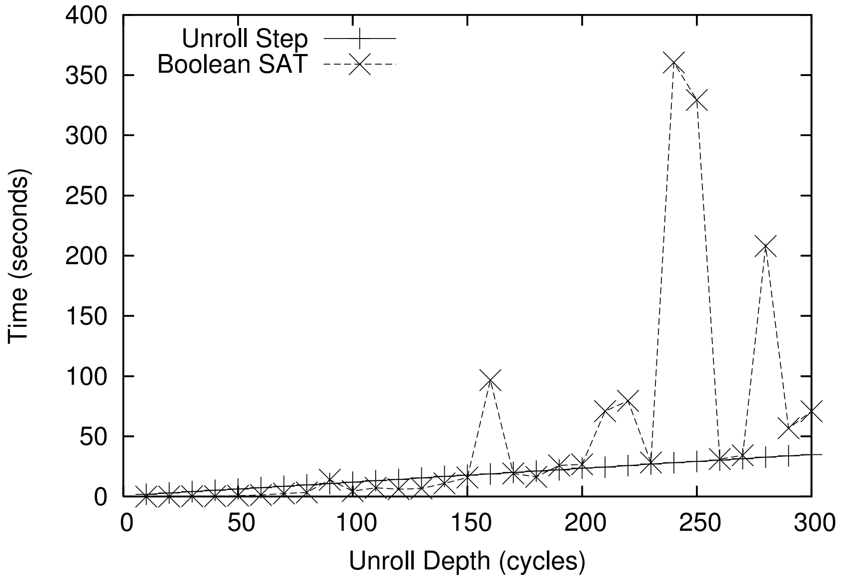

Figure 6-5: Time to perform Design Comparison on RS232 UART circuit

likely due to (1) the nature of solving NP-complete problems, and (2) the method in which the optimized SAT solver relsat approached each task.

6.4.2 PIC16F84 8-bit Microcontroller

The next set of tests were run on the 4 Trojans implemented on PIC16F84 8-bit microcontrollers. The PIC16f84 represents a useful benchmark representing small microcontrollers, and the feasibility of detecting malicious modifications in such chips. During testing, all four Trojans were flagged during the unrolling process for suspicious nonstandard logic. It was not necessary to run the Boolean SAT comparison, as all four Trojans were identified prior to that point. The unrolling process provides a rigorous method for identifying nonstandard logic that is used in hardware Trojan modifications, despite that type of detection not being its main purpose.

6.4.3 Wishbone Conmax IP Core

To fully test the Design Comparison technique, larger circuits were necessary. We chose the OpenCores Wishbone Conmax Interconnect Matrix IP Core as the benchmark, because it is orders of magnitude larger than the other circuits. Of the three test benchmarks, there

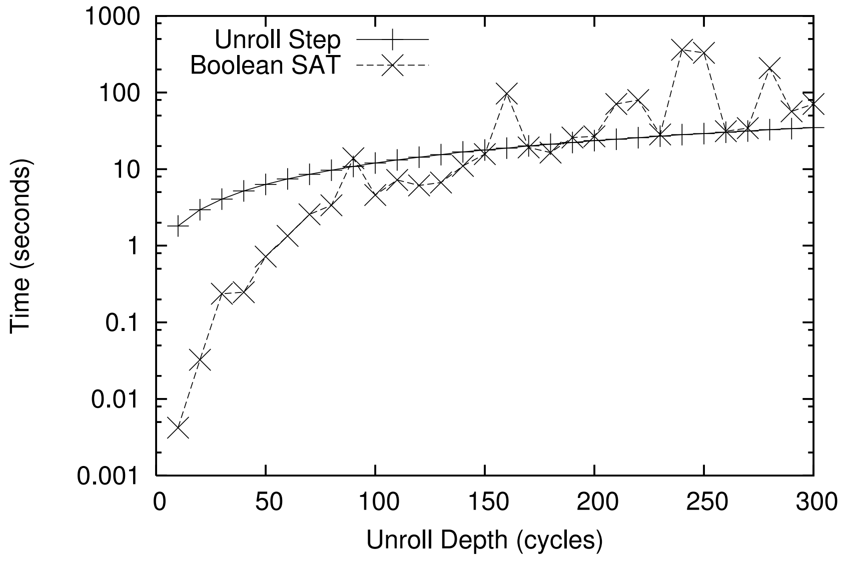

Figure 6-6: Time to perform Design Comparison on RS232 UART circuit on a logarithmic scale

was also a large difference in the implementation of the different benchmarks. This was most pronounced when re-synthesizing the designs to the OSU 45-nm standard cell library and comparing their sizes. Compared to the first benchmark, the second two were roughly 10% smaller in area (126,986 um^2 for the first, and 114,160 um^2 for the other two). They were also structured differently, as the first version of the benchmark contained built-in scan-chain functionality, while the latter two benchmark Trojans did not.

Among the three Trojan benchmarks, the second Trojan (T200) was flagged during the unrolling process due to suspicious circuitry. The other two Trojans (T100 and T300) continued on for comparison, and were quickly identified even with a very short unroll depth (with a window of 5 cycles).

6.4.4 ISCAS '89 Benchmarks

The final set of tested benchmarks are grouped together, as they are all from the ISCAS89 sequential benchmark test suite. There was a total of four benchmark circuits tested, with 10 total Trojans.

- s15850 - 1 Trojan

- s35932 - 3 Trojans
- s38417 - 3 Trojans
- s38584 - 3 Trojans

These benchmarks are all significantly larger than the RS232 UART, with the last three being roughly 1/3rd of the size of the Wishbone Conmax core. These circuits exhibited more variability than the previous benchmarks.

First, four Trojans (the s15850 Trojan and all three s38584 Trojans) were detected during the comparison process with no difficulty. The three s38584 Trojans were detected regardless of what combination of benchmark and Trojan were used. Since all three Trojans had different payloads, they were functionally different from each other. This difference was what was detected during comparison. Furthermore, as seen in Fig. 6-7, the cost in time to unroll and test for satisfiability remained similar to those results from the RS232 circuit for the unroll depth of up to 50 cycles.

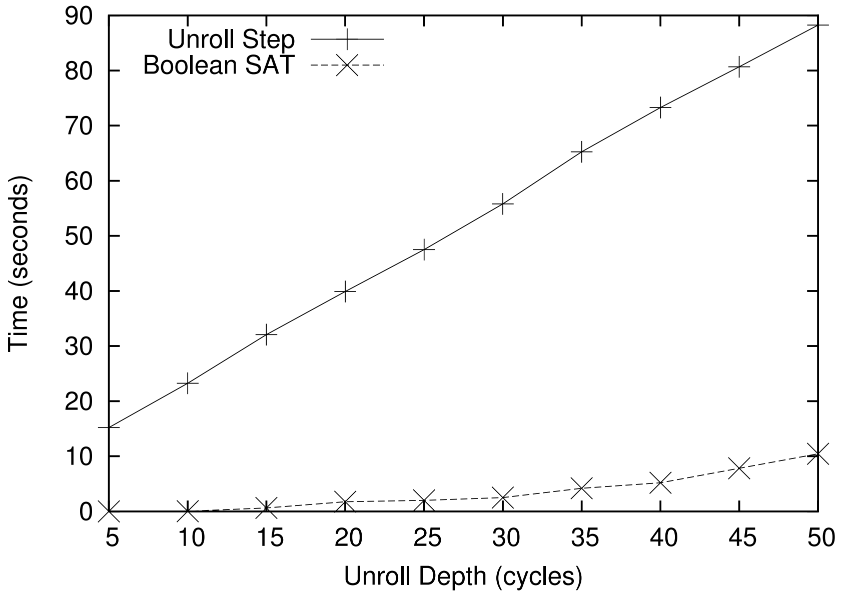

Figure 6-7: Time to perform Design Comparison on ISCAS'89 s15850

However, the tests on the s38417 benchmarks showed false negatives, despite possessing circuitry that would certainly raise flags during unrolling. This was noticed when the s38417-T300 Trojan, which possesses a ring oscillator with 29 stages, was not flagged for cyclic logic. Under further examination, it was discovered that the malicious components in this circuit were completely removed when Cadence RTL-Compiler elaborated and flattened the design. As a result, instead of a false negative, the result was correctly determined: a true negative.

Finally, the test on the s35932 benchmark showed positive results: (1) the T100 circuit was flagged for having strangely crafted clock logic, (2) T200 was correctly identified via comparison, and (3) T300 was determined to have no Trojan. Further examination showed that again, RTL-compiler had completely removed the Trojan in T300, leaving a clean circuit (and thus a false negative). T100 was flagged because a comparator was built into it using an unclocked flip-flop. As this logic was not working on the same clock as the rest of the circuit, the unrolling process identified it as suspicious.

6.4.5 Limitations

Several key limitations were determined by these experiments, the most significant of which was inherent in the tools used for testing, namely SMV, AIGER, and to a small part RTL-Compiler and its overzealous optimization. In addition, limitations with the unroll depth and SAT variation were identified.

6.4.5.1 Tools

The use of SMV was the most significant limiting factor, as it greatly restricted the circuits that could be tested. While it worked well for many of the circuits, a large number of the other Trojans available on Trust-HUB were untestable due to SMV. For these circuits, the initial use of SMV would cause a segmentation fault and fail simply due to the size of the circuit. For more robust testing of larger circuits, an alternative to SMV is necessary.

The second limiting factor was in the AIGER toolset's interaction with SMV. Again, this implementation of SMV cannot handle large circuits well, so the SMV conversion to and-invert graphs would occasionally fail for circuits unrolled to great depths. This was likely the reason why the s38417-T100 and T200 Trojans went undetected, i.e., the unroll depth was limited by the toolset and could not be extended much at all.

The final limiting factor in the tools is that the Cadence RTL compiler has an efficient optimization algorithm. Even when it is not instructed to optimize the design for space or timing, it removes unnecessary circuitry. While this is technically a good thing for general circuit design, it makes Trojan testing more difficult. For instance, the TrustHUB benchmarks include four Trojans inserted into a BasicRSA cryptographic circuit, but upon analysis, all four Trojans were completely removed by RTL compiler prior to analysis. It is effective enough to say that rigorous synthesis is an effective way of removing Trojans relying on delicate signals and structures.

6.4.5.2 Unroll Depth

A key limitation inherent in the Design Comparison technique is the unroll depth. While this was accentuated by the SMV tools faulting when testing deep unroll depths, a Trojan with an exceedingly long activation time could remain hidden simply by exceeding the test window. This Trojan would require a large amount of additional clocked circuitry to allow for the delay, such that it would be difficult to hide it from other techniques that focus on identifying suspicious structures. Since Design Comparison is designed specifically to be used in combination with such other techniques, there is low risk that a Trojan would be missed by both detection mechanisms.

6.4.5.3 SAT Variation

As seen with the RS232 testing and with the analysis of the s38584-T100 Trojan, the cost in time to run the NP-complete Boolean satisfiability problem can be unreliable. While the time to solve such a problem increases steadily with the unroll depth, occasional tests take much longer to run than would normally be predicted based upon circuit size, circuit complexity, and unroll depth. In order to combat this challenge, it might be more efficient to use multiple SAT-solving processes in parallel, testing with different techniques simultaneously, such as stochastic algorithms [81] and Davis-Logemann-Loveland [82] algorithms.

6.5 Summary

Through testing, the overwhelming majority of Trojans were detected through implementation of Design Comparison. The only Trojans circuits that escaped detection were ones that were eliminated by tools during circuit synthesis (and were therefore correctly assessed as "not Trojan" circuits). However, most of the tested Trojans did not make it past the unrolling process to the design comparison process, as they were flagged early on due to cyclical logic, suspicious clocks, or other nonstandard circuitry. For future work, a larger variety of Trojans, that do not rely on such easily detected structures, would be helpful in demonstrating the efficacy of Design Comparison. The other key tasks to accomplish as future work includes overcoming some of the limitations covered. The most critical limitation to address is the tools used and the inherent inability of the AIG tools to analyze the largest circuits. Despite these limitations; however, this process has proven effective at detecting malicious functionality that is hidden within third-party intellectual property. Considering the ease of implementing this test, and the resulting damage from failing to identify something suspicious included in a design, this process can be implemented on high-risk applications that might be vulnerable to catastrophic failure if hidden functionality were present.

Chapter 7

Assessment of Trojan Impact on Host Circuitry

While the previous chapter explores security pre-fabrication, this chapter and Chap. 8 focus on security post-fabrication. This chapter explores the full impact that hardware Trojans can have upon a host circuit, in terms of change to its layout area, dynamic power consumption, and ambient leakage power. Interestingly, this dissertation finds that Trojans have a customizable impact, allowing for a malicious designer to invest time and resources to make an attack more difficult to detect. With eighteen different Trojans that all performed the same type of attack on the same circuit, there was a large variety of different impacts observed in terms of area and power consumption. This problem is expanded when examining multiple circuits, where 90% of Trojans had an impact on layout area of less than 2.5%, an impact on dynamic power of less than 6%, and an impact on leakage power of less than 3.3%. These changes are an order of magnitude smaller than the inherent randomness inflicted upon the circuit by the fabrication processes. To make matters worse, half of the Trojans tested were ten-times smaller, with an average impact on area of less than 0.18%, an impact on dynamic power consumption of less than 0.7%, and an impact on leakage power of less than 0.17%.

Furthermore, this chapter also shows that there is a negative correlation between circuit size and a Trojan's impact upon the circuit in terms of percent area and power. This confirms that compromising larger circuits does not necessarily require larger Trojans. As the percent change due a Trojan decreases with circuit size, Trojans get progressively more difficult to identify as circuit size increases.

The key contributions from this chapter are:

- An extensive characterization of area and power for the hardware Trojans from the Trust-Hub benchmark suite using several fabrication technologies,
- Correlation analysis for different hardware Trojan effects within the same design and across different designs, and
- Analysis of absolute change in relation to both process variation and detection range.

7.1 Benchmark Analysis

Among the many classifications provided by the Trojan taxonomy in Fig. 3-2, possibly the most interesting classification is that of Trojan effect, or *payload*. These payloads generally fall into four categories: (1) Denial-of-Service, (2) Leaking Data, (3) Degrading Performance, and (4) Changing Functionality. A Trojan with a denial-of-service (DOS) payload cripples the design, making it unusable. A remote kill-switch falls under this category. A Trojan with a payload of leaking data could be inserted into some circuit with secret information, such as a cryptographic circuit, to leak the hidden key. Such an attack would surreptitiously subvert the security of such a chip. A Trojan that degrades performance would be one that does not completely cause the chip to fail, but still impairs functionality in a meaningful way, reducing reliability. Finally, Trojans that change functionality modify the host circuit in a way as to add functionality that was originally not present in the circuit, such as a remote hijack.

With such a large variety of possible Trojan functions, quantifying the impact of a hardware Trojan becomes difficult. The amount of circuitry necessary to implement a of kill switch is likely to be significantly less than the amount necessary to implement a surreptitious remote hijack. As detection techniques rely upon differences in side-channel signals such as power consumption, determining whether Trojans are likely to have a significant impact is absolutely necessary.

7.1.1 Benchmark Suite Composition

To identify the impact of Trojan circuits on a host design, we examine more than 60 Trojans in the Trust-Hub repository. While this repository contains a significant number of Trojans, they are not all represented at the same abstraction level. Also, 4 Trojans subverting the MC8051 microcontroller were included from a previous study [30]. To effectively compare the results between the many different Trojans, it was necessary to limit analysis to those Trojans that were supplied in VHDL or Verilog, and compatible with Synopsys Design Compiler and Cadence RTL Compiler. Furthermore, some of the Trojans supplied, such as the wide variety of RS232 Trojans, did not have an accompanying *Trojan-Free* version, making it difficult to accurately measure the impact that the Trojan logic has on the original circuit.

7.1.2 Methodology

Analysis of the Trojans was performed by synthesizing these hardware descriptions to standard cell libraries using industry standard software. Synopsys Design Compiler (DC) [83] was used to synthesize the designs to the Synopsys 28/32-nm and 90-nm standard cell libraries, and Cadence RTL Compiler (RC) [84] was used to synthesize the designs to the Oklahoma State University 45-nm FreePDK standard cell library [85, 86]. The Trojan-free circuits were then synthesized to these libraries for comparison. Power estimates were obtained through Cadence RC and Synopsys DC. These synthesized designs were then compared with the clean circuit to identify the impact of Trojan circuitry on the original design. Both Cadence and Synopsys tools were applied to reduce the impact of tool-specific artifacts. By comparing the percent change based on each tool, three characterizations of each Trojan were created. While in some cases these characterizations were similar to one another, they often had different results due to the manner in which the tools interpreted the Trojan circuits. In this way, there were effectively three times as many Trojan circuits analyzed.

7.2 Impact Within a Design

There are many different types of Trojans, and each type can be implemented in a significantly different manner. To fully analyze the impact of a Trojan on a host circuit, it is necessary to reduce the variation in the results caused by such differences. This analysis is performed by examining eighteen Trojans leaking sensitive information in an Advanced Encryption Standard (AES) 128-bit cryptographic circuit. While each of these Trojans have roughly the same payload, each Trojan delivers the payload in a different way. Although optimization was minimized for both synthesis tools, some of the Trojans without logic-based payloads were effectively removed as unnecessary logic. These Trojans were removed from the results, and the overall impact on area for the remaining Trojans is shown in Fig. 7-1.

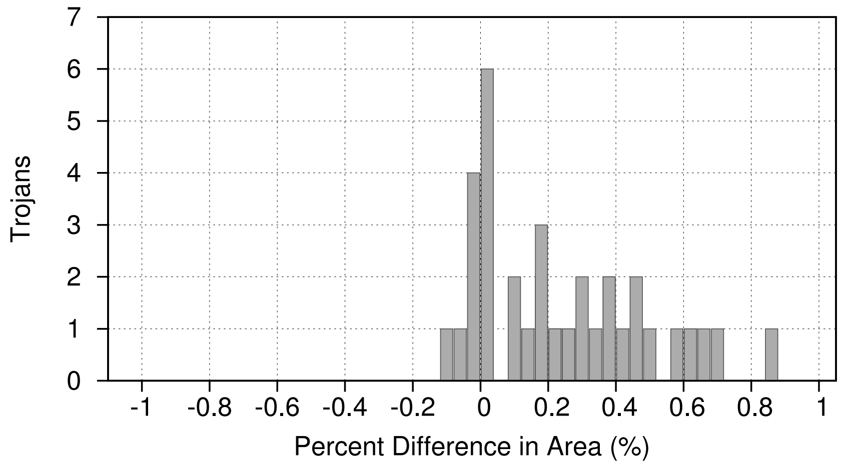

Figure 7-1: Change in percent area due to data leak Trojans in AES cryptographic circuits. Calculated on the 32-nm, 45-nm, and 90-nm standard cell libraries.

One of the key results to note is that the overall impact of the Trojans on the area of the AES cryptographic circuit is very small, at only 0.25%, with a standard deviation of 0.23%. For a circuit that requires 556,851 μm, 692,119 μm, and 1,588,698 μm for the 32-nm, 45-nm, and 90-nm libraries respectively, this result represents a very small change that could be very difficult to detect. Another interesting result is the wide range over which these Trojans influence the AES circuit. The vast majority have an impact around 0.1%, but the results spread from -0.2% to 1% change in area. This range represents a surprisingly large amount of variety for Trojans with the same classification on the same circuit.

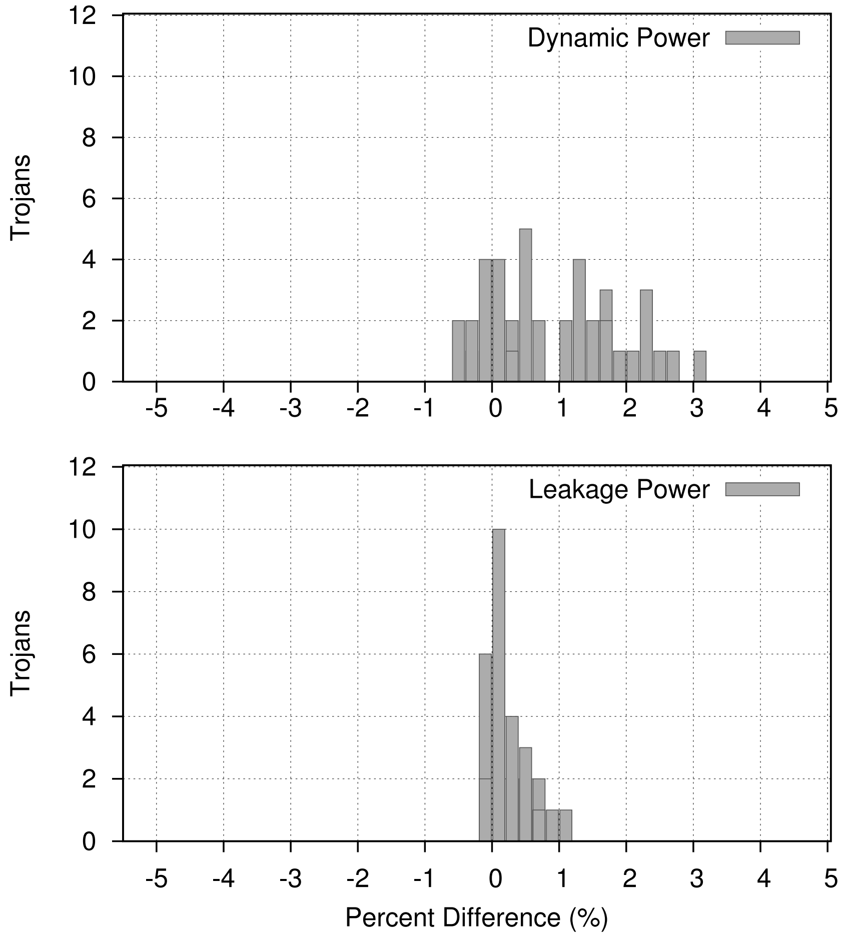

Figure 7-2: Change in percent power due to data leak Trojans in AES cryptographic circuits. Calculated on the 32-nm, 45-nm, and 90-nm standard cell libraries.

The analysis of dynamic power and leakage power (Fig. 7-2) shows similar results to those for area. The key difference is that the Trojans have a larger general impact on the dynamic power consumption of the circuit, with an average value of 0.96% and a standard deviation of 0.97%. However, note that this impact ranges from -0.5% to 3.1%, with a remarkably even spread across this range. The results for leakage power show almost the exact same trends as the results for area.

One of the key findings is that, for a single type of Trojan on a single circuit, the implementation of the Trojan is highly customizable. The variety of implementations available to the Trojan designer can greatly affect the final impact that the Trojan will have on a circuit's performance. While the average impact on dynamic power was around 1%, there were several implemented Trojans that exhibited significantly smaller absolute differences. In fact, 42% of the synthesized Trojan circuits influenced the dynamic power consumption of the AES circuit by 0.5% or less.

7.3 Impact Across Different Designs

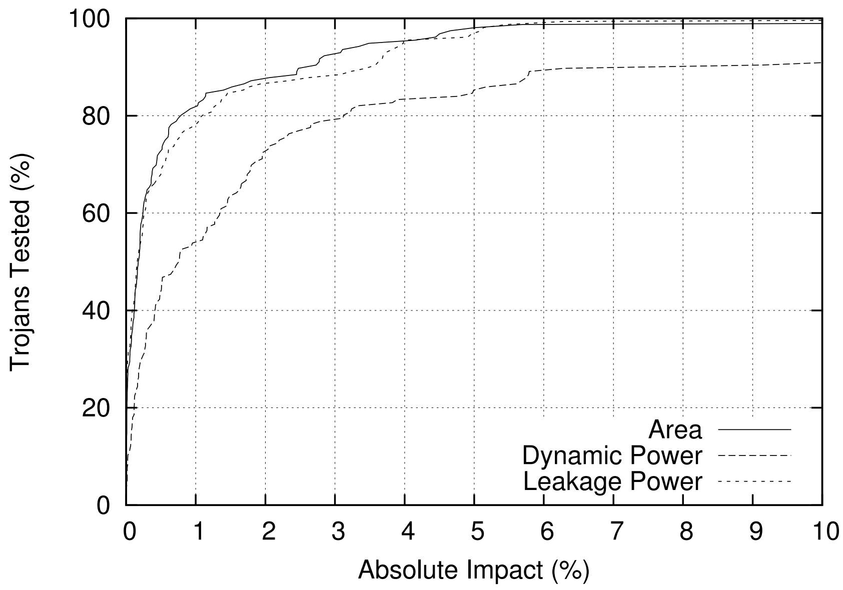

Figure 7-3: Cumulative distribution of absolute impacts on area, dynamic power consumption, and leakage power on all benchmark circuits.

Unfortunately, the data leak Trojan is not the only type of Trojan that can be inserted into a circuit, and the AES cryptographic block is not the only circuit that can be subverted by malicious hardware. Almost any type of circuit can be subverted if attacked, and there are many different ways that Trojans can be implemented. A large variety of circuits modified with Trojans were tested using the same methodology used to test the AES circuits (Section 7.1.1). A cumulative distribution for these impacts is shown in Fig. 7-3. These results are visualized as as a cumulative distribution to identify the portion of tested Trojans that had impacts below a certain amount. For instance, even if a detection technique proves that it is able to identify a change in area of 1%, only 20% of the Trojans tested had an impact of this size or greater.

7.3.1 Impact on Area

When examining the overall impact on area, the results were startling. The overwhelming majority of the Trojans had a very small impact on the area of their host-circuits, with 80% of the Trojans providing an impact of less than 1%, and 95% of Trojans affecting the area of their host circuit by less than 4.5%. Interestingly, some of the Trojans caused a net decrease in area, visible in Fig. 7-4.

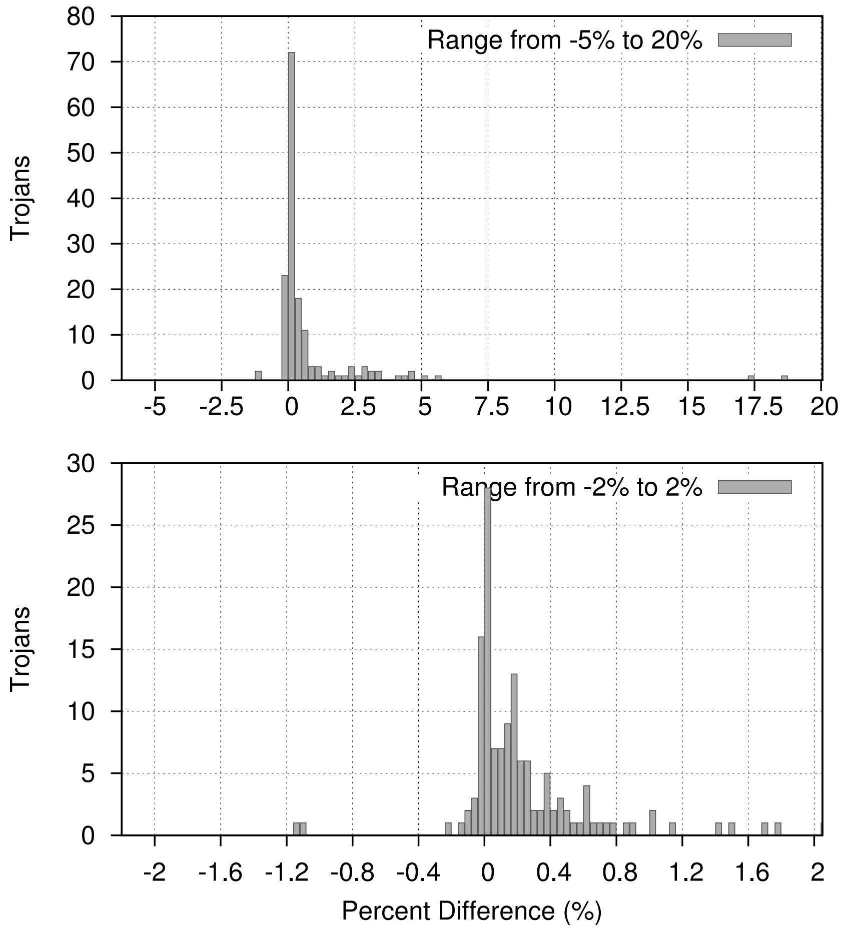

Figure 7-4: Change in percent area over all tested Trojans. Calculated across the 32-nm, 45-nm, and 90-nm standard cell libraries. The bottom is a zoomed-in version of the top plot.

Table 7-1: Average impact on area for each host circuit.

Circuit	Average	Deviation
AES	0.25%	0.23%
b19	1.38%	1.40%
EthernetMAC10GE	0.002%	0.003%
MC8051	0.18%	0.60%
PIC16F84	4.95%	4.02%
s15850	0.36%	0.49%
s35932	0.05%	0.12%
s38417	0.12%	0.10%
s38584	2.54%	6.08%
vga_lcd	0.009%	0.007%
wb_conmax	0.46%	0.48%

Table 7-2: Average impact on area for each Trojan classification.

Classification	Average	Deviation
All	0.81%	2.28%
Leak Information	0.42%	0.83%
Denial of Service	1.28%	2.80%
Change Functionality	0.64%	2.26%

One of the key factors influencing the total Trojan impact on area was the host circuit being attacked. After analysis, there was noted a significant difference in area required to implement Trojans within each individual circuit circuit. The average impact on the host area for each circuit is shown in Table 7-1.

For some of the tested circuits, such as in the PIC16F84 microcontroller, the impact on the area was identified to be incredibly large, averaging at a 4.95% change in area to the whole circuit. For other circuits, the change in area was less than one-tenth of a percent difference.

Unfortunately, it is not possible to completely segregate circuit type and Trojan type, as not all Trojan types are equally represented across all of the benchmarks. For instance, a Trojan that leaks information is useless in a circuit that does not handle information that could be considered secret. The Trojans were grouped based on their effect, ignoring the host circuit, and the averages are shown in Table 7-2.

7.3.2 Impact on Power

Measuring the overall change in power consumption in the Trojan circuits provided further interesting results. The leakage power observed closely mirrored the results found for the change in area, and is shown in Fig. 7-5. However, dynamic power showed significantly divergent results compared to the area (Fig. 7-6).

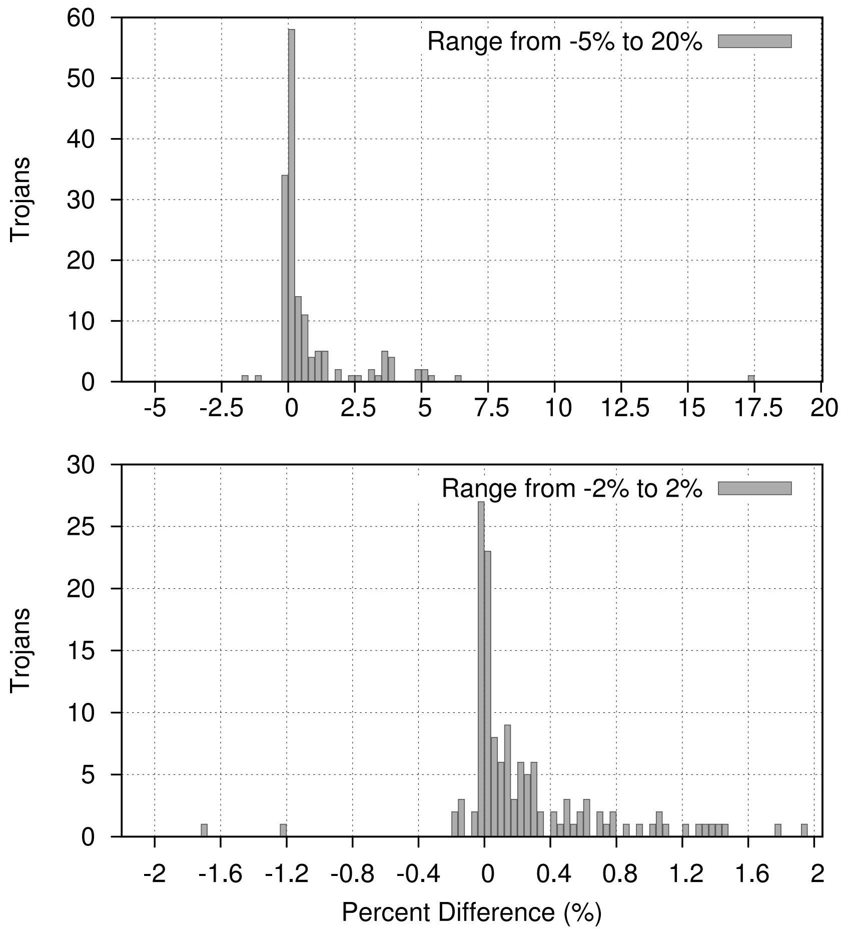

Figure 7-5: Change in leakage power consumption over all tested Trojans. Calculated on the 32-nm, 45-nm, and 90-nm standard cell libraries. The bottom is a zoomed-in version of the top plot.

The impact of the Trojan circuitry on the dynamic power consumption of the circuits showed a significantly wider spread, ranging from a -25% decrease in power to a near 90% increase in power, shown in Fig. 7-6. However, even ignoring these outliers, there is still a wider spread compared to the area, as only 54% of the Trojans have an absolute impact of less than 1%. The tail of the result has spread out slightly, so that 80% of the Trojans have an absolute impact of 3.2% or less on the dynamic power consumption of the circuit. By breaking down the results based upon individual circuits, the most extreme results can

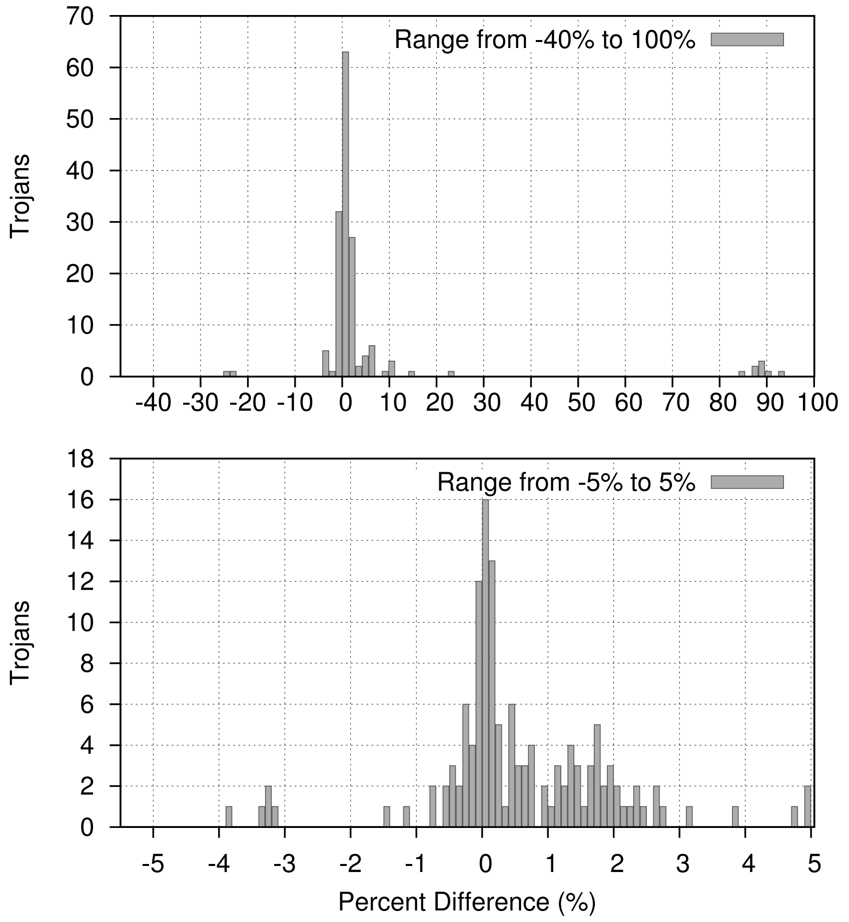

Figure 7-6: Change in dynamic power consumption over all tested Trojans. Calculated on the 32-nm, 45-nm, and 90-nm standard cell libraries. The bottom is a zoomed-in version of the top plot.

quickly be identified as coming from several of the Trojans in the PIC16F84 microcontroller. Table 7-3 displays the average change in dynamic power and the standard deviation based upon each host circuit. Throughout all of the circuits, both the average difference and the standard deviation of the difference are consistently larger than those seen for impact on area.

Although the impact due to Trojans varies greatly depending upon the host circuits, these benchmarks are also very different circuits in terms of area, performance, and struc-

Table 7-3: Average impact on dynamic power for each host circuit.

Circuit	Average	Deviation
AES	0.96%	0.97%
b19	-3.00%	8.43%
EthernetMAC10GE	0.38%	0.88%
MC8051	1.79%	4.02%
PIC16F84	61.51%	40.56%
s15850	4.67%	9.03%
s35932	2.01%	2.52%
s38417	1.50%	1.72%
s38584	1.07%	2.49%
vga_lcd	-0.06%	0.05%
wb_conmax	0.74%	2.23%

ture. The largest gap is between the PIC16F84 microcontroller and the 128-bit AES cryptographic block; on the 32-nm standard cell library the PIC16F84 required an area of 6,290 μm^2, while the AES block required an area of 556,851 μm^2. The question then becomes how much the Trojan impact is dependent upon the size of the host circuit. Fig. 7-7 shows the plot of the average absolute difference in area vs. the size of the circuit. There is a

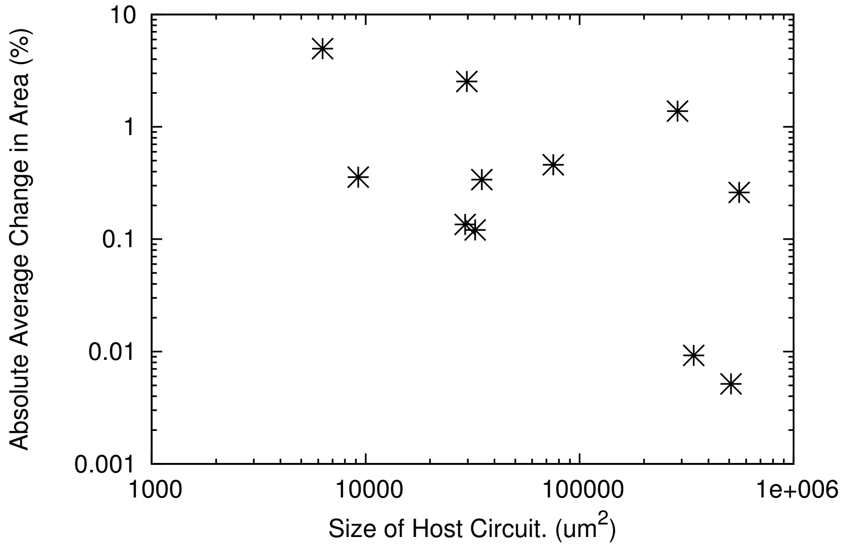

Figure 7-7: Average absolute change in area due to Trojans compared to the size of the host circuit when synthesized to the 32-nm standard cell library.

general trend present in these results, correlating the impact on percent area with the size of the compromised circuit. While the largest impact is seen on small host circuits, and the smallest impact is similarly seen on large host circuits, the circuits in-between are slightly less clear-cut. To more accurately portray these results, the Pearson product-moment correlation coefficient (Pearson's r) [87] is calculated for each of these results (Fig. 7-8). For this coefficient, a Pearson's r of 1 would represent a total positive correlation between the two results, while a -1 would represent a total negative correlation. The middle ground of -0.1 to +0.1 represents no correlation between the two variables. Overall, the Pearson's r

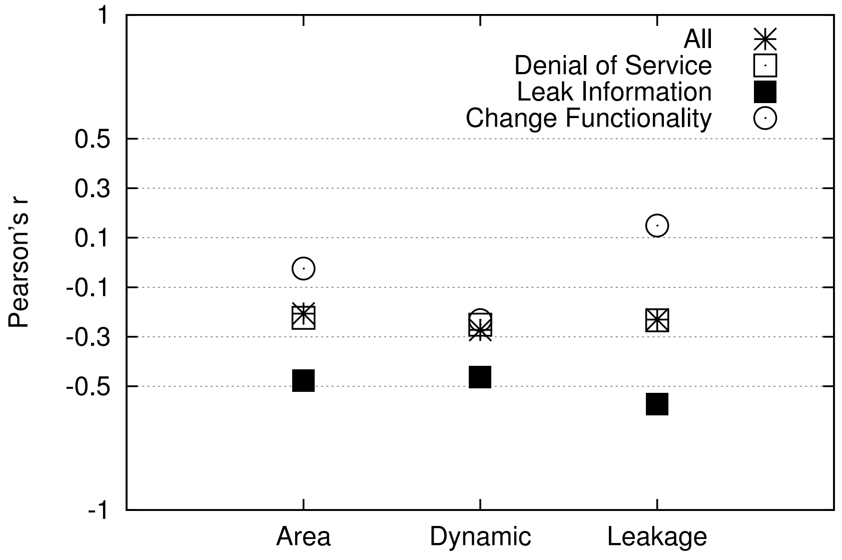

Figure 7-8: Pearson product-moment correlation coefficient relating Trojan impact on percent area, dynamic power, and leakage power to the size of the modified circuit when synthesized to the 32-nm library.

calculated for these Trojans show a relatively consistent, small negative correlation between Trojan impact and host circuit area. This negative correlation is increased for Trojans that leak information, and reduced for Trojans that change functionality. These results can be interpreted to mean that as circuits get larger, the percent impact on area and power due to denial of service and data leak Trojans decreases on average. Trojans that change functionality show no correlation with host circuit area.

7.3.3 Limitations

There are two main limiting factors for this research. The first, and most significant limitation is the Trojans that were used in these tests. While more than 60 Trojans were tested to obtain these results, there was a large amount of variation between these circuits. Furthermore, some circuits were more thoroughly represented than others. These tests could always benefit from more Trojans for analysis in order to verify the results.

The second limiting factor was the tools and libraries used in this experiment. Although both Synopsys Design Compiler and Cadence RTL-compiler were used, there are other tools available that might display different results. Furthermore, even within DC and RC, there are many different ways to synthesize a design. These tests were run with a minimum amount of optimization effort from the tools, in order to preserve Trojan presence. Changing this effort might change the impact of each Trojan on their host circuits.

7.4 Summary

Securing a hardware design is not a simple process. There are many possible points at which a design may be compromised and detection is almost always difficult. This is even more the case with regards to Trojans that are inserted during fabrication of the IC. From this research, there are three key findings:

(1) Trojans have large variability within a design, even when performing the same functionality. This flexibility means that detection techniques need to aim to catch the smallest spectrum of Trojans, as a malicious attacker could easily invest resources into making an attack significantly harder to identify. There was no easily discernible minimum change that could be used as a goalpost for detection techniques. This result significantly raises the bar as to what represents rigorous detection.

(2) The average Trojan impact was quite small. While the impact of an individual Trojan circuit on a benchmark circuit in terms of area and power can vary to great degrees, we can say that the general change is around 1% or less. This result means that even if a detection technique is validated against one or two Trojans, there is little meaning to such a detection, as individual Trojan characteristics can change by a large amount and the

tested Trojan could represent an outlier. An example of this was seen by the PIC Trojan that increased dynamic power consumption by 90% over the original circuit's consumption. Whether or not a technique can identify such a Trojan is unhelpful, as there are plenty of ways to identify if something is wrong when a circuit is consuming almost double the power it should be. Instead, techniques should be validated against several Trojans, or against a specific measured size of impact.

(3) The general problem of detecting and identifying Trojans is likely to grow in difficulty as the circuit size increases. A Trojan detection technique that can barely identify a circuit in a small circuit is unlikely to scale this success to a larger one.

Overall, the task of detecting malicious hardware post-fabrication is extremely difficult. For many chip designers, security through trusted fabrication plants might be the preferred method of eliminating this risk. For others, obfuscating the original chip should be explored as a way of increasing the cost of inserting an attack into a design.

Chapter 8

Identifying Malicious Hardware through Software Parameter Modelling

One of the problems faced in identifying unknown modifications within circuits has been forming an effective basis for what represents "suspicious". Many experiments make use of a physical "golden copy" that is obtained through some unknown process. Unfortunately, obtaining such a convenient chip is not always possible; physically fabricating another chip at a trusted fabrication plant is expensive, and is likely to have different trends in power and timing than the unknown chips. Even different lots from the same fabrication plant can have widely differing process variation, interfering with Trojan detection. Another option would be to arbitrarily pick a single chip to treat as a golden copy, and then later send this chip off for reverse-engineering. Unfortunately, this requires using extremely high-detail imaging techniques which can be extremely expensive, especially when examining chips at the size of modern fabrication processes.

As a solution, in this chapter we introduce a method to create a substitute golden copy from the expected circuit layout based on the design sent for fabrication. This chip will vary greatly from the measurements from the physical chip; however, this substitute golden copy and the physical untrusted chip will be subject to the same series of input test vectors. After each test, the parameters of the components within the substitute golden copy are modified based on the inputs vectors and the corresponding differences observed between the transient responses of the two chips. The goal of this modification is to minimize the difference in the transient responses between the two chips. This process is shown in Fig.

8-1.

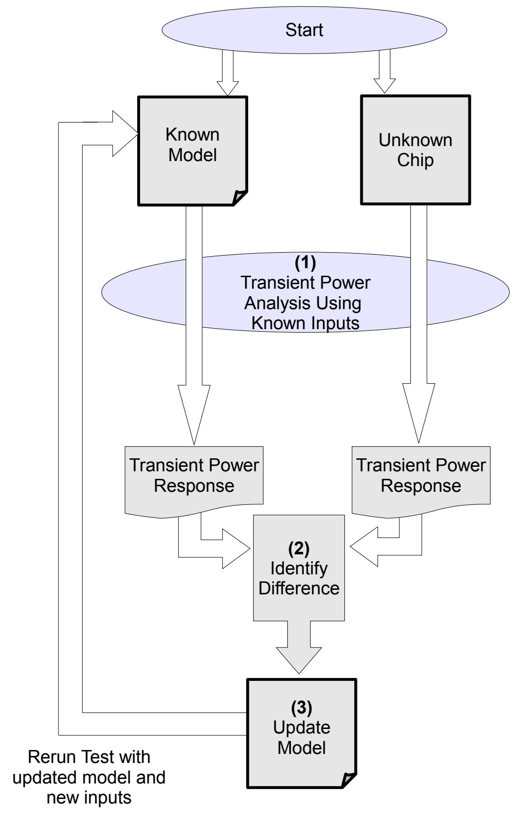

Figure 8-1: Overview of model matching process.

After this training cycle has continued for a large number of tests, the internal parameters of the substitute golden copy represents useful metrics for identifying suspicious trends in the untrusted chip. Even if the Trojan circuitry is never triggered in the untrusted chip, this circuitry is going to affect the transient power response of the chip. However, as such components do not exist in the ideal model that forms the substitute golden copy, the only way that the training can approximate this unknown functionality is through large modifications to the parameters of the known components. The component parameters for the substitute golden copy therefore represent an estimate for how much the untrusted chip varies from the expected design. If these parameters fall within an expected amount of variation, then the chip can be designated as trustworthy. Similarly, if the parameters show strange trends that are unlikely to be caused by process variation alone, then the chip is designated as untrustworthy and can be examined in further detail.

As with many classification techniques that attempt to identify suspicious behavior, the exact use of this technique will depend upon the degree of trust desired by the testers. This can be shown as a cost-benefit comparison between maximizing true positives to prevent attacks on the design, and minimizing false positives to minimize yield loss (shown in Fig. 8-2).

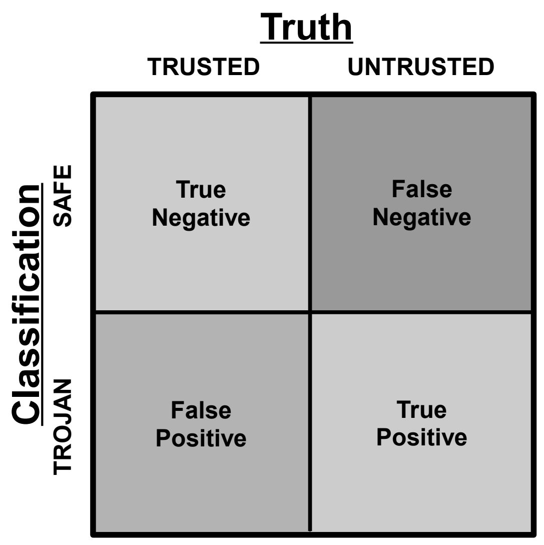

Figure 8-2: Classification of unknown chips.

For an application with a high cost of failure, the threshold for classifying a chip as untrustworthy would be much lower than a casual application with a lower risk.

The key contributions from this chapter are:

- Development of a technique for identifying very small modifications to a design by matching variations in a real design to a digital design, bypassing masking caused by intra-die variations in transistor parameters.
- Testing of this technique over a case study of a 74181 ALU and 32 small modifications that have no functional impact on the circuit, only adjusting the dynamic power consumption of the circuit by between 0% and 3%.

8.1 Testing Process

To fully verify the effectiveness of this process, a case study was implemented through transient testing in Cadence Spectre Circuit Simulator [74] using the Oklahoma State University 45-nm FreePDK standard cell library [80]. As this was entirely modeled in software, the untrusted chips were represented in testing by unknown software netlists constructed with large amounts of intra-die variation, as well as possible malicious Trojans. These unknown chips were then treated as physical chips and used to train known models.

The circuit used for testing was an implementation of the ISCAS 74181 Arithmetic Logic Unit constructed using 392 transistors. Each transistor was assigned a unique model, and parameters were adjusted for each model individually. This granularity had a negative effect on the optimization processes used by the Spectre Simulator to determine transient power consumption, resulting in unnecessarily long simulation time. However, comprehensive tests on the 74181 were still possible.

There were three main parts to the experimental process:

- Process variation modeling and application
- Representing Trojan circuitry/malicious hardware in the 74181
- The training process for the known circuit

8.1.1 Process Variation

The key obstructing factor when determining the trustworthiness of a chip is the process variation inherent in the chip. As this variation changes between chips as well as between transistors, the model for this process variation has a key impact on the efficacy of testing. The key parameters that are adjusted by the inherent variation are transistor gate length and threshold voltage. While other parameters will vary in small amounts, their impact is eclipsed by the impact of changes in gate length and threshold voltage.

For every transistor, the model parameters were varied for threshold voltage and gate length based on a Gaussian distribution with a maximum change capped at three times the standard deviation, to limit outliers. To best determine the amount of variation to incorporate, 50 samples were generated at each level of variation, and tested to determine average impact on power consumption within the circuit. These results, shown in Fig. 8-3, tend to show that using a Gaussian distribution with a standard deviation of 3% across the parameters is sufficient to mask the impact of roughly 90% of the Trojans characterized earlier in this thesis.

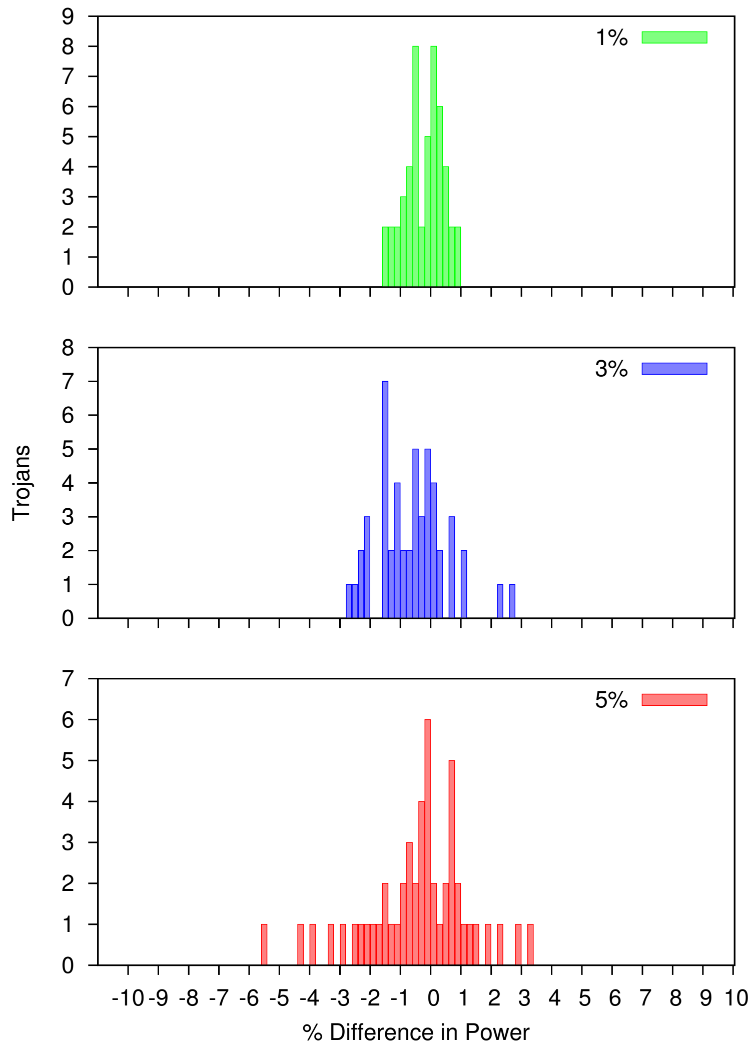

Figure 8-3: Impact observed testing 50 circuits with randomly distributed variation within each circuit.

8.1.2 Representing a Trojan Circuit

As there are no good examples of a Trojan inserted into a 74181 ALU, it was necessary to develop some modification to the circuit that can take the place of a Trojan. However, as the circuit is relatively small, large insertions could have an inordinate impact on power. The most important restriction of such Trojan models is that the impact on side-channel signals fit the same impacts determined in Chapter 7.

After testing several models, it was determined that increasing the gate length of a transistor by 200% would generally cause changes in power consumption in the 74181 equal to those determined by Trojans in larger circuits.

The Trojans tested were then selected based on their impact measured, ensuring that none of the Trojans had an impact on transient power consumption beyond that seen by process variation.

8.1.3 Training

The last, and arguably most important part of the modeling cycle is the actual method in which the known circuit is trained to the unknown circuit. The key goal here is to minimize the difference in transient power consumption between the two circuits, and component parameters in the known circuit are modified to reduce the difference between the two transient measurements.

The first step is to measure transient power consumption of both the known and unknown circuits under the exact same input vectors. As inputs are one of the few variables that are under control of testers, it is important to ensure that all circuits are always tested under the same set of input test vectors. For these tests, the full sequence of test vectors was randomly generated before testing began, and then continually used throughout testing. The transient power was then compared at each change in inputs to determine how much they differ for each input. At this point, two key pieces of information can be determined: (1) The transistors active during each change in inputs, and (2) The difference in power consumption for each input.

The second step in the training process requires modifying each active transistor in the known circuit to more closely approximate the response observed by the unknown circuit. In this case, the active transistors are determined by stepping through the circuit with the input vector and identifying any point at which a transistor will change states, even if it is only temporary. Any transistor that changes states at any point is considered active, and has gate length and threshold voltage modified based on transient results. However, there is only one measurement (power consumption), so there is little meaning to using multiple parameters to represent changes in both threshold voltage and gate length. Instead, a separate value representing deviation from the base was recorded for each transistor, which was applied to determine what the transistor gate length and threshold voltage should be. To ensure these parameters were accurately addressed by this variable, gate length and threshold voltage were adjusted individually in an inverter to determine their scope of impact. The impact is shown in Fig. 8-4.

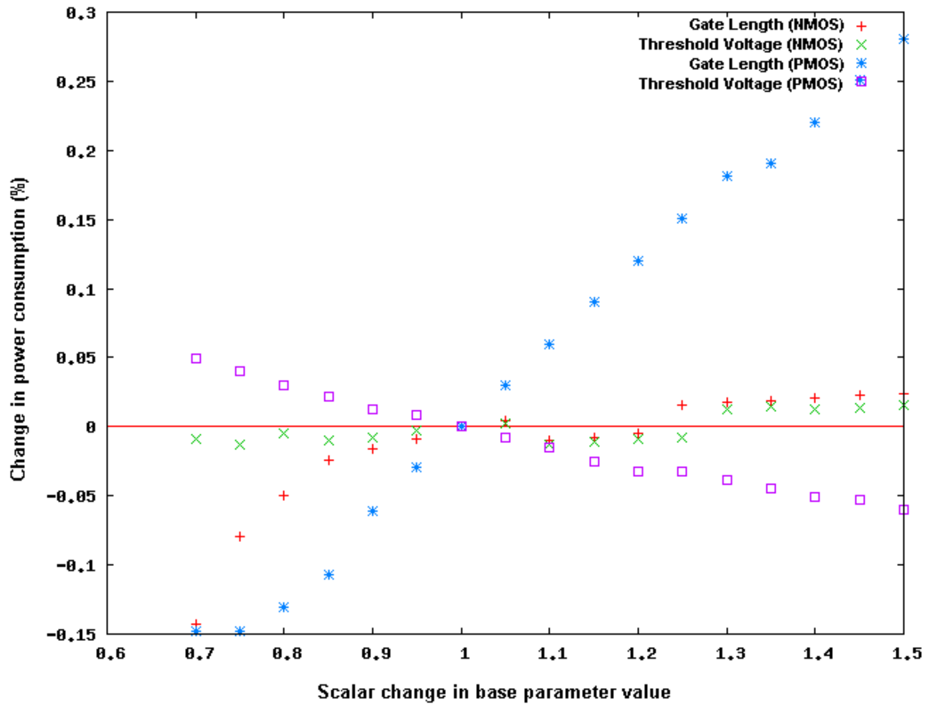

Figure 8-4: Percent change in power consumption of transistor dependent upon change in threshold voltage and gate length of PMOS and NMOS transistors.

Based on these measurements, we determined first, that the NMOS transistors have a significantly lower impact on power consumption than the PMOS transistors, and second, that the gate length and threshold voltage of PMOS transistors relate to power consumption in opposite ways, with the gate length having a larger impact on threshold voltage. Therefore, whenever the known circuit consumed less power than the unknown circuit over a specific input, the gate length was increased for relevant PMOS transistors and threshold voltage was decreased. The amount of this change was directly proportional to the observed difference between the known and unknown transient responses. Repeatedly putting the known circuit through this process gradually minimizes the differences observed between the known circuit and unknown circuit. The exact proportion is unimportant, as almost any value still results in eventual convergence. The main limitation is that a small proportion might take longer to reach convergence, and a large proportion will overshoot, and result in ringing.

8.1.4 Algorithm

For clarity, a pseudocode implementation of the training process is included below.

```
for Iterations 1 to 100 do
    for Random inputs 1 to 20 do
        Identify active transistors
        Run transient power analysis on Unknown
        Run transient power analysis on Known model
    end
    for Each transistor in [Known model] do
        if Active in at least 1 input then
            for Each active input do
                Identify difference (Unknown - Known)
                Multiply difference by scalar $K_L$ for change in gate length
                Multiply difference by scalar $K_T$ for change in threshold voltage
                  $K_L$ and $K_T$ are arbitrary constants based on size of circuit.
                  Small values converge slowly, large values converge quickly
            end
            Calculate average change in gate length over all inputs
            Calculate average change in threshold voltage over all inputs
            if PMOS then
                Add average change to gate length of transistor
                Subtract average change from threshold voltage of transistor
            end
            if NMOS then
                Subtract average change from gate length of transistor
                Add average change to threshold voltage of transistor
            end
        else
            Ignore transistor - do not change any parameters
        end
    end
end
```

8.2 Results

To verify that this analysis technique can identify Trojan circuits, several transistors were selected, each representing different a Trojan. A total of 32 Trojans were then selected based upon their impact measured. Of these Trojans, 25 had an impact on transient power of less than 1%, and 8 had an impact of less than 0.1%. Note that most of the transistors in the circuit had a much larger impact than these 32 when corrupted, causing as much as a 17% change in transient power consumption.

After increasing the size of these transistors by 200%, 10 copies of each Trojan circuit were created with 3% Gaussian variation added to all transistors in addition to the Trojan transistor. Furthermore, each copy was created separately, with its own unique set of variations. These 320 circuits, along with 60 circuits with no Trojan transistors were used as "unknown" circuits for training over 100 cycles using the same set of randomly chosen input vectors.

After this training process was completed, the digital copies were then examined. Parameters for every transistor in each copy were modified during the training process, depending upon the results of each transient simulation. These changes could be quantified as a percentage of the original, unchanged value (e.g., 100% representing no change, 110% representing a 10% increase, and 90% representing a 10% decrease). This list of changes represents the primary information gained from this technique, as a transistor with a large change represents a large deviation between the unknown circuit and the trained model.

For an unknown circuit with no Trojan circuitry, the expected deviation between the known and unknown circuits will be small. However, for an unknown circuit with Trojan circuitry, the model has a more difficult time matching values, as the two circuits are structurally different. This difficulty will be embodied as larger changes in the parameters of the known, trained model.

To simplify analysis, the geometric mean of all of the transistor differences was taken for each trained circuit. This gives a single value that represents the overall difference observed in the trained circuit. If we compare the Trojan circuits with the clean circuits using a one-tailed t-test, we can identify whether or not they are observably different. The results from these tests are shown in Table 8-1. These p-values represent the likelihood of the null hypothesis, i.e., that the Trojan values and the clean values are inseparable. A low p-value represents a rejection of the null hypothesis. Based upon these p-values, 25 out of 32 are below 0.10, 23 out of 32 are below 0.05, and 19 out of 32 are below 0.01. In all cases, the lower the p-value, the more clearly defined the two groups are.

It is important to point out that for the last 5 Trojans (with impacts of 1.58%, 1.90%, 2.04%, 2.63%, and 3.01%) the trained transistor values were far beyond those of other tests. For example, the maximum change observed in a transistor for Trojan #31 was consistently around 170%, while the largest transistors trained to the clean circuits were consistently below 105%. This trend continued in some of the smaller Trojans. For instance, the largest observed change in transistors when training to Trojan #4 was around 109%.

One of the most interesting results here is that some of the Trojans (such as Trojan #25 with an impact of 0.86%) were undetectable, when compared to Trojans with much smaller impacts. One of the possible reasons behind this behavior is likely due to the individual location of the Trojan modification. For instance, a modification on a highly active path will cause a greater change in power over more inputs, masking possible Trojans signs.

8.2.1 Limitations

Although this process has detected extremely small changes in the two designs, there are several key limitations of this research.

(1) the lack of a golden copy greatly increases the difficulty of determining exactly what point is “suspicious”. As this test was performed using a digital case-study some confounding variables were not addressed, such as transient power differences caused by the chip packaging and wires. Many confounding variables were accounted for through the extremely large intra-die process variation, but such changes might mask Trojan differences. This directly ties into (2) the fact that this test was performed using a digital representation of a physical chip, instead of an actual physical chip. There is no doubt that using a physical chip for this test would be more significant of a result than using digital chips. Finally, (3) the 74181 is an extremely small circuit. While this research did not test larger circuits, the Trojans were also proportionately smaller, making up for much of the difference. However,

Table 8-1: Measured Trojan impacts on transient power consumption and p-value for one-tailed t-test

Trojan	Impact (%)	p-value
1	0.01%	0.434
2	-0.05%	0.482
3	-0.05%	**0.017**
4	0.06%	**0.000**
5	0.06%	**0.000**
6	-0.06%	**0.000**
7	0.07%	**0.017**
8	0.08%	**0.000**
9	-0.13%	**0.001**
10	-0.13%	**0.001**
11	-0.17%	0.300
12	0.20%	0.330
13	0.21%	**0.027**
14	-0.22%	**0.002**
15	-0.25%	0.389
16	-0.29%	**0.000**
17	-0.34%	**0.009**
18	0.37%	**0.000**
19	0.37%	**0.000**
20	0.48%	**0.000**
21	0.52%	0.470
22	-0.54%	**0.005**
23	-0.67%	**0.067**
24	0.70%	**0.000**
25	0.86%	0.386
26	1.07%	**0.010**
27	-1.40%	**0.063**
28	1.58%	**0.000**
29	1.90%	**0.000**
30	2.04%	**0.000**
31	2.63%	**0.000**
32	3.01%	**0.000**

in the end it would be better to fully test this technique on larger circuits as well.

8.3 Summary

Overall, the model matching process is shown to be effective at identifying very small changes in a circuit through process variation. Detecting a 200% change in gate length of a single transistor in a circuit masked by process variation across all components is a very difficult task, especially when the variation between components on the chip is as high as a 3% Gaussian standard deviation. As these impacts are equivalent in size to the Trojans observed in Chapter 7, detecting these Trojans would pose the same difficulty.

Further work in this area foremost involves reworking much of the test method to optimize the testing process in order to reduce simulation time, (which was around 25 seconds per iteration for the 74181). This might require using a different toolset, as there are other tools by Cadence and other vendors that might be more efficient for simulating low-level differences between transistor models in a larger circuit. Once this performance barrier has been overcome, this technique can be readily used on larger circuits, both in a digital and physical environment.

Another area to explore further is to incorporate a more detailed analysis of the parameter values. The results presented in Table 8-1 demonstrate a noticeable difference between Trojan and non-Trojan circuits, despite the analysis relying on a single value: the geometric mean. Analysis based on clustering or other techniques could yield a more clear distinction between circuits.

Chapter 9

Summary

As modern integrated circuits grow increasingly complex in both structure and function, the risk of hidden malicious functionality cannot be ignored. While cybersecurity and attacks on software designs are rampant and make headlines, the production path of integrated circuits provides vectors for attacking the hardware designs as well. This risk is compounded with every new iteration of fabrication technology, pushing the cost of building cutting-edge fabrication plants higher and higher. The goal of this dissertation is to reduce this risk, and provide developers a safe alternative beyond relying solely on trusted IP providers and trusted fabrication plants.

The contributions from this dissertation can be broken into three main parts: (1) the development of a comparison technique for validating the safety of a digital hardware design of a circuit without obtaining a known safe copy; (2) the assessment of impacts on area and power of a large variety of Trojan circuits obtained using multiple tools and technologies; and (3) the development of a technique to assign a level of trust to an unknown chip without using destructive imaging or relying upon an expensive known-safe chip.

9.1 Detection in third-party IP

As one of the most overlooked areas, hidden malicious hardware in third-party IP presents a real threat to modern designs. With the current climate where IP vendors who are suspicious of chip designers lock down their design to prevent theft, chip designers have fewer ways to verify that the blocks they are using perform only as intended. This research provides an alternative to chip designers, allowing to identify any unknown functionality that might be hidden within the design. Although this functionality could be from a digital watermark hidden in the chip to protect the vendor, it could just as easily come from Trojan circuitry.

9.2 Assessment of Trojan impacts on area and power

While there has been an increasing amount of research on hardware Trojans in the past few years, the actual process of verifying that a technique can identify a Trojan is unclear. There are no ways to compare the efficacy of techniques, and in many cases, chip designers must rely upon researchers that the technique is effective. This dissertation has explored a variety of Trojans and established how much of an impact that a large variety of Trojans, providing an assortment of functionality, can have on integrated circuits. For some techniques, the impact on area will be more important, as rearranging circuitry to provide for Trojan functionality can hint at the presence of Trojans. For other techniques, the Trojan impact on power represents how easily the Trojan can be detected by techniques relying upon side-channel power analysis. Furthermore, this assessment provides a baseline measurement for Trojan stealth, as a Trojan with a small impact relative to other Trojans is unlikely to be found, while a Trojan with a large impact is not going to be effective at demonstrating the performance of a new technique.

9.3 Identification of Trojans in fabricated chips

The final portion of this research provides a method for chip designers to verify that the chips they have received from an overseas foundry are trustworthy. Furthermore, this method is inexpensive, as it does not require costly imaging, destruction of chips, nor fabrication of additional chips to represent a safe group. As this technique only relies upon EDA tools and standard testing tools, it could be performed during standard logical verification.

9.4 Opportunities for Future Work

This research showed the effectiveness of techniques for identifying Trojans in third-party IP and in fabricated circuits. However, there are many possible areas in which this research can be expanded. This section explains how this dissertation could be extended further.

First, the design comparison technique proposed by this research can be explored with larger and more complicated circuits. This could be achieved by testing a variety of toolsets to determine which can handle the process the fastest, and would be easiest to offer to designers for verifying design blocks.

Second, the assessment of Trojan impacts can be further explored by analyzing more Trojans, eventually developing a set of benchmarks containing large variety of extremely stealthy Trojans. One of the problems with the Trust-Hub benchmarks lie in the fact that many of the Trojans rely upon the same circuitry and, therefore identification of often results in identification of the complete set. By developing a larger group of Trojan circuits, future Trojan detection techniques can be more accurately evaluated, allowing for chip designers to make knowledgeable decisions about what techniques to implement.

Third, the parameter modeling method for identifying Trojans in fabricated chips can be further refined in two ways: (1) testing the process on other tools, to allow for testing of larger circuits in a reasonable period of time, and (2) physically fabricating several test chips to determine the minimum granularity detectable. Although the testing does require measurement of the transient power response, this does not require expensive test equipment to perform. The primary limitation of this approach lies in how effectively the model can be built to match the physical design, and then matched through training.

Fourth, another area in which this research could be further expanded is through the development of an anti-Trojan toolkit for designers. The primary goal would be two-fold: (1) to increase awareness of the threat, and therefore reduce the number of designers making unwise decisions and therefore introducing possible threats to modern chips, and (2) to provide easy and inexpensive options to designers that are unable to afford trusted fabrication plants and trusted vendors for all of their external resources. By providing easy testing processes to these designers, this security hole could be almost entirely closed.

References

[1] Semiconductor and Manufacturing Design Community, http://semimd.com/blog/2012/02/10/umc-seeks-to-shed-image-as-'fast-follower'/.

[2] D. Collins, "DARPA Trust in IC's Effort (BRIEFING CHARTS)," 2007.

[3] S. Adee, "The hunt for the kill switch," *IEEE Spectrum*, vol. 45, no. 5, pp. 34–39, 2008.

[4] S. T. King, J. Tucek, A. Cozzie, C. Grier, W. Jiang, and Y. Zhou, "Designing and implementing malicious hardware," in *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats (LEET '08)*, 2008.

[5] R. Torrance and D. James, "The state-of-the-art in semiconductor reverse engineering," in *48th ACM/EDAC/IEEE Design Automation Conference (DAC '11)*, 2011, pp. 333–338.

[6] L. Spitzner, "Honeypots: Catching the insider threat," in *19th Annual Computer Security Applications Conference.* IEEE, 2003, pp. 170–179.

[7] US Department of Justice, VisionTech Components Press Release, http://www.justice.gov/usao/dc/news/2011/oct/11-472.html.

[8] J. Stradley and D. Karraker, "The electronic part supply chain and risks of counterfeit parts in defense applications," *IEEE Transactions on Components and Packaging Technologies*, vol. 29, no. 3, pp. 703–705, 2006.

[9] P. Godefroid, J. van Leeuwen, J. Hartmanis, G. Goos, and P. Wolper, *Partial-order methods for the verification of concurrent systems: an approach to the state-explosion problem.* Springer Heidelberg, 1996, vol. 1032.

[10] W. Donath and H. Ofek, "Automatic identification of equivalence points for boolean logic verification," *IBM Technical Disclosure Bulletin*, vol. 18, no. 8, pp. 2700–2703, 1976.

[11] J. Roth, "Hardware verification," *IEEE Transactions on Computers*, vol. C-26, no. 12, pp. 1292–1294, 1977.

[12] O. Coudert, C. Berthet, and J. C. Madre, "Verification of synchronous sequential machines based on symbolic execution," in *Automatic verification methods for finite state systems.* Springer, 1990, pp. 365–373.

[13] O. Coudert, J. C. Madre, and C. Berthet, "Verifying temporal properties of sequential machines without building their state diagrams," in *Proceedings of the 2nd International Workshop on Computer Aided Verification*, ser. CAV '90. London, UK: Springer-Verlag, 1990, pp. 23–32. [Online]. Available: http://dl.acm.org/citation.cfm?id=647759.735162

[14] D. K. Probst and H. F. Li, "Using partial-order semantics to avoid the state explosion problem in asynchronous systems." in *CAV*, ser. Lecture Notes in Computer Science, E. M. Clarke and R. P. Kurshan, Eds., vol. 531. Springer, 1990, pp. 146–155. [Online]. Available: http://dblp.uni-trier.de/db/conf/cav/cav1990.html#ProbstL90

[15] D. M. Chickering, D. Geiger, D. Heckerman *et al.*, "Learning bayesian networks is np-hard," Citeseer, Tech. Rep., 1994.

[16] J. R. Burch, E. M. Clarke, K. L. Mcmillan, D. L. Dill, and L. J. Hwang, "Symbolic model checking: 10 20 states and beyond," in *Proceedings of the 5th Symposium on Logic in Computer Science*, 1971, pp. 428–439.

[17] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Transactions on Computers*, vol. C-35, no. 8, pp. 677 –691, 1986.

[18] A. Valmari and T. Jokela, "Embedded software validation through state space generation," in *Second International Conference on Software Engineering for Real Time Systems, 1989*, 1989, pp. 278–282.

[19] P. Godefroid and P. Wolper, "A partial approach to model checking," in *Proceedings of Sixth Annual IEEE Symposium on Logic in Computer Science, (LICS '91),*, 1991, pp. 406–415.

[20] J. Roth, "Diagnosis of automata failures: A calculus and a method," *IBM Journal*, Jul. 1966.

[21] K.-H. Tsai, J. Rajski, and M. Marek-Sadowska, "Star test: the theory and its applications," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 9, pp. 1052–1064, 2000.

[22] R. Dandapani and S. Reddy, "On the design of logic networks with redundancy and testability considerations," *IEEE Transactions on Computers*, vol. C-23, no. 11, pp. 1139 – 1149, nov. 1974.

[23] A. F. M. Abramovici, M. A. Breuer, "Digital systems testing and testable design," *Computer Science*, 1990.

[24] K.-T. Cheng and V. Agrawal, "A partial scan method for sequential circuits with feedback," *IEEE Transactions on Computers*, vol. 39, no. 4, pp. 544 –548, apr 1990.

[25] S. Narayanan, R. Gupta, and M. Breuer, "Configuring multiple scan chains for minimum test time," in *IEEE/ACM International Conference on Computer-Aided Design, (ICCAD-92)*, nov, 1992, pp. 4 –8.

[26] ——, "Optimal configuring of multiple scan chains," *IEEE Transactions on Computers*, vol. 42, no. 9, pp. 1121 –1131, sep 1993.

[27] M. C. Kidd, "Self-test of digital evaluation equipment," *IEEE Transactions on Aerospace*, vol. 1, no. 2, pp. 1283 –1296, 1963.

[28] A. Avizienis, G. Gilley, F. Mathur, D. Rennels, J. Rohr, and D. Rubin, "The star (self-testing and repairing) computer: An investigation of the theory and practice of fault-tolerant computer design," *IEEE Transactions on Computers*, vol. C-20, no. 11, pp. 1312 – 1321, nov. 1971.

[29] Aeroflex-Gaisler, http://www.gaisler.com/cms/.

[30] T. Reece, D. B. Limbrick, X. Wang, B. T. Kiddie, and W. H. Robinson, "Stealth assessment of hardware trojans in a microcontroller," in *IEEE 30th International Conference on Computer Design (ICCD '12)*, 2012-Oct. 3, pp. 139–142.

[31] S. Skorobogatov and C. Woods, "Breakthrough silicon scanning discovers backdoor in military chip," in *Cryptographic Hardware and Embedded Systems (CHES '12)*, E. Prouff and P. Schaumont, Eds. Springer Berlin Heidelberg, 2012, vol. 7428, pp. 23–40.

[32] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware trojans," *Computer*, vol. 43, no. 10, pp. 39–46, 2010.

[33] F. Wolff, C. Papachristou, S. Bhunia, and R. Chakraborty, "Towards Trojan-Free trusted ICs: Problem analysis and detection scheme," in *Design, Automation and Test in Europe (DATE '08)*, 2008, pp. 1362–1365.

[34] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental security analysis of a modern automobile," in *2010 IEEE Symposium on Security and Privacy (SP)*, 2010, pp. 447–462.

[35] A. Abdel-Hamid, S. Tahar, and E. M. Aboulhamid, "IP watermarking techniques: Survey and comparison," in *The 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications, 2003*, 2003, pp. 60–65.

[36] M. Lewandowski, R. Meana, M. Morrison, and S. Katkoori, "A novel method for watermarking sequential circuits," in *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST '12)*, 2012, pp. 21–24.

[37] G. Becker, M. Kasper, A. Moradi, and C. Paar, "Side-channel based watermarks for integrated circuits," in *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST '10)*, 2010, pp. 30–35.

[38] D. Ziener, F. Baueregger, and J. Teich, "Multiplexing methods for power watermarking," in *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST '10)*, 2010, pp. 36–41.

[39] E. Clarke, O. Grumberg, and D. Long, "Verification tools for finite-state concurrent systems," in *A Decade of Concurrency Reflections and Perspectives.* Springer, 1994, pp. 124–175.

[40] G. S. May and C. J. Spanos, *Fundamentals of semiconductor manufacturing and process control.* Wiley. com, 2006.

[41] A. Waksman and S. Sethumadhavan, "Tamper evident microprocessors," in *IEEE Symposium on Security and Privacy (SP '10)*, 2010, pp. 173–188.

[42] K. J. Berk, "The impact of IEEE-1076 on VHDL (hardware description language)," No. AFIT/GCE/ENG/88D-1. AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING, Tech. Rep., 1988.

[43] J. Soden and R. Anderson, "IC failure analysis: techniques and tools for quality reliability improvement," *Proceedings of the IEEE*, vol. 81, no. 5, pp. 703 –715, may 1993.

[44] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitecture," in *40th Annual Design Automation Conference (DAC '03)*, 2003, pp. 338–342.

[45] S. Borkar, "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation," *IEEE Micro*, vol. 25, no. 6, pp. 10–16, 2005.

[46] P. Song, F. Stellari, D. Pfeiffer, J. Culp, A. Weger, A. Bonnoit, B. Wisnieff, and M. Taubenblatt, "Marvel malicious alteration recognition and verification by emission of light," in *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST '11),*, 2011, pp. 117–121.

[47] D. Nedospasov, J.-P. Seifert, A. Schlosser, and S. Orlic, “Functional integrated circuit analysis,” in *Hardware-Oriented Security and Trust (HOST), 2012 IEEE International Symposium on*, 2012, pp. 102–107.

[48] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, “Trojan detection using IC fingerprinting,” in *IEEE Symposium on Security and Privacy (SP)*, 2007, pp. 296–310.

[49] X. Wang, M. Tehranipoor, and J. Plusquellic, “Detecting malicious inclusions in secure hardware: Challenges and solutions,” in *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST ’08).*, 2008, pp. 15–19.

[50] R. Rad, J. Plusquellic, and M. Tehranipoor, “Sensitivity analysis to hardware trojans using power supply transient signals,” in *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST ’08)*, 2008, pp. 3–7.

[51] M. Banga and M. Hsiao, “A region based approach for the identification of hardware trojans,” in *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST ’08)*, 2008, pp. 40–47.

[52] ——, “A novel sustained vector technique for the detection of hardware trojans,” in *22nd International Conference on VLSI Design, 2009*, 2009, pp. 327–332.

[53] ——, “VITAMIN: voltage inversion technique to ascertain malicious insertions in ICs,” in *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST ’09)*, 2009, pp. 104–107.

[54] J. Aarestad, D. Acharyya, R. Rad, and J. Plusquellic, “Detecting trojans through leakage current analysis using multiple supply pad iddqs,” *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 4, pp. 893–904, Dec 2010.

[55] X. Wang, H. Salmani, M. Tehranipoor, and J. Plusquellic, “Hardware trojan detection and isolation using current integration and localized current analysis,” in *IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems (DFTVS ’08)*, 2008, pp. 87–95.

[56] R. Rad, J. Plusquellic, and M. Tehranipoor, "A sensitivity analysis of power signal methods for detecting hardware trojans under real process and environmental conditions," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 12, pp. 1735–1744, DECEMBER 2010.

[57] J. Li and J. Lach, "At-speed delay characterization for IC authentication and trojan horse detection," in *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST '08)*, 2008, pp. 8–14.

[58] ——, "Negative-skewed shadow registers for at-speed delay variation characterization," in *25th International Conference on Computer Design (ICCD '07)*, 2007, pp. 354–359.

[59] D. Rai and J. Lach, "Performance of delay-based trojan detection techniques under parameter variations," in *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST '09)*, 2009, pp. 58–65.

[60] Y. Jin and Y. Makris, "Hardware trojan detection using path delay fingerprint," in *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST '08)*, 2008, pp. 51–57.

[61] J. Sulistyo and D. Ha, "A new characterization method for delay and power dissipation of standard library cells," *VLSI Design*, vol. 15, no. 3, pp. 667–678, 2002.

[62] F. Koushanfar and A. Mirhoseini, "A unified framework for multimodal submodular integrated circuits trojan detection," *IEEE Transactions on Information Forensics and Security*, vol. 6, pp. 162–174, 2011.

[63] R. Chakraborty, S. Paul, and S. Bhunia, "On-demand transparency for improving hardware trojan detectability," in *IEEE International Workshop on Hardware-Oriented Security and Trust(HOST '08)*, 2008, pp. 48–50.

[64] J. Valamehr, T. Sherwood, R. Kastner, D. Marangoni-Simonsen, T. Huffmire, C. Irvine, and T. Levin, "A 3-d split manufacturing approach to trustworthy system development," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 4, pp. 611–615, 2013.

[65] F. Emeson, A. Emtenan, G. Siddharth, and M. V. Tripunitara, "Securing computer hardware using 3d integrated circuit (IC) technology and split manufacturing for obfuscation," in *Proceedings of the 22nd USENIX Security Symposium.* USENIX Association, 2013.

[66] J. Rajendran, O. Sinanoglu, and R. Karri, "Is split manufacturing secure?" in *Design, Automation Test in Europe Conference Exhibition (DATE '13)*, 2013, pp. 1259–1264.

[67] A. Avizienis and J. C. Laprie, "Dependable computing: From concepts to design diversity," *Proceedings of the IEEE*, vol. 74, pp. 629–638, 1986.

[68] A. Avizienis, J. C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, pp. 11–33, 2004.

[69] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 10–25, 2010.

[70] T. Reece, D. B. Limbrick, and W. H. Robinson, "Design comparison to identify malicious hardware in external intellectual property," in *IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom '11)*, 2011, pp. 639–646.

[71] R. S. Chakraborty, F. Wolff, S. Paul, C. Papachristou, and S. Bhunia, "Mero: A statistical approach for hardware trojan detection," in *Cryptographic Hardware and Embedded Systems (CHES '09).* Springer, 2009, pp. 396–410.

[72] S. Shazli and M. Tahoori, "Using boolean satisfiability for computing soft error rates in early design stages," *Microelectronics Reliability*, vol. 50, no. 1, pp. 149–159, Jan. 2010. [Online]. Available: http://www.sciencedirect.com/science/article/B6V47-4XBP9F3-1/2/98138eaae38c25d812871f78661699a4

[73] M. Hicks, M. Finnicum, S. King, M. Martin, and J. Smith, "Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically," in *IEEE Symposium on Security and Privacy (SP '10)*, 2010, pp. 159–172.

[74] Cadence Design Systems, http://www.cadence.com/.

[75] AIGER, http://fmv.jku.at/aiger/.

[76] relsat, http://code.google.com/p/relsat/.

[77] Cadence SMV, http://www.kenmcmil.com/smv.html.

[78] Open Source Hardware IP-Core Community, http://opencores.org/.

[79] Trust-HUB, https://trust-hub.org/.

[80] Oklahoma State University: 45-nm FreePDK standard cell library, http://vlsiarch.ecen.okstate.edu/flow/.

[81] B. Selman, H. Kautz, and B. Cohen, "Local search strategies for satisfiability testing," in *Proceedings of the Second DIMACS Challange on Cliques, Coloring, and Satisfiability*, 1993.

[82] M. Davis, G. Logemann, and D. Loveland, "A machine program for theorem proving," *Communications of the ACM*, vol. 5, pp. 394–397, 1962.

[83] Synopsys University Program, Academic Tools Available: www.synopsys.com/Community/UniversityProgram/Pages/default.aspx.

[84] Cadence Worldwide University Software Program, Available: www.cadence.com/support/university/pages/default.aspx.

[85] J. Grad and J. E. Stine, "A standard cell library for student projects," in *IEEE International Conference on Microelectronic Systems Education (MSE '03)*, June 2003, pp. 98–99.

[86] J. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. Davis, P. Franzon, M. Bucher, S. Basavarajaiah, J. Oh, and R. Jenkal, "Freepdk: An open-source variation-aware design kit," in *IEEE International Conference on Microelectronic Systems Education (MSE '07)*, June 2007, pp. 173–174.

[87] J. Lee Rodgers and W. A. Nicewander, “Thirteen ways to look at the correlation coefficient,” *The American Statistician*, vol. 42, no. 1, pp. 59–66, 1988.