A Multi-Parameter Functional Side Channel Analysis Method for Hardware Trojan Detection in Untrusted FPGA Bitstreams

Christopher William Bell
University of South Florida, cwbell@Mail.usf.edu

Follow this and additional works at: http://scholarcommons.usf.edu/etd
Part of the Computer Engineering Commons

Scholar Commons Citation

This Thesis is brought to you for free and open access by the Graduate School at Scholar Commons. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact scholarcommons@usf.edu.
A Multi-Parameter Functional Side Channel Analysis Method for Hardware Trojan Detection in Untrusted FPGA Bitstreams

by

Christopher W. Bell

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Engineering Department of Computer Science and Engineering College of Engineering University of South Florida

Major Professor: Srinivas Katkoori, Ph.D.
Swaroop Ghosh, Ph.D.
Jay Ligatti, Ph.D.

Date of Approval: March 18, 2013

Keywords: Security, Trust, FPGA, ASIC, Fabrication

Copyright © 2013, Christopher W. Bell
Dedication

This thesis is dedicated to my family. I couldn’t have accomplished any of this without their love and support. My mother and father supported me throughout the entire chaotic journey from high school to here, put up with every delay and complication, and most importantly never lost faith in my abilities. I love you.

In memory of my Grandfather, John Fogarty, Manhattan Borough Commander, F.D.N.Y. (Ret.) Loving husband, father and grandfather. He will always be loved and missed.
Acknowledgments

Matthew Lewandowski and Richard Meana worked with me as a team in the Embedded Systems Challenge. Matt helped with designing and testing the control systems and with tying the system together. He also discovered the clock problems with one of the designs that caused our test to fail, saving an absolute ton of time. Richard was the only member of our team that was able to articulate complete sentences during the ESC presentation, and he is the only reason we participated in the competition in the first place. This never would have been finished without them.

The support and feedback provided by Dr. Katkoori were instrumental in every part of this thesis development. He patiently dealt with me as I missed deadline after deadline. He introduced us to the field of hardware security as well as the competition. He had confidence in our abilities despite having no prior knowledge of anything remotely security related.

Special thanks to the Polytechnic Institute of New York University for hosting the Embedded System Challenge, especially the event coordinators Ramesh Karri and Jeyavijayan Rajendran. Without them we may not have ever pursued this area of research.
Table of Contents

List of Tables iii

List of Figures iv

Abstract vi

1 Introduction and Background 1
   1.1 Thesis Organization 2
   1.2 Hardware Trojan Horse 3
   1.3 Trojan Taxonomies 3
   1.4 Trojan Design 3
      1.4.1 Effects 4
      1.4.2 Activation 5
      1.4.3 Implementation 6
   1.5 FPGA Considerations 6
   1.6 Chapter Summary 8

2 Proposed Dynamic Multi-Parameter Analysis Technique 9
   2.1 Hardware Trojan Detection Techniques - A Survey 9
      2.1.1 Functional Verification 9
      2.1.2 Side-Channel Analysis 11
      2.1.3 Power Analysis 11
      2.1.4 Thermal Profiling 13
      2.1.5 Gate-Level Characterization 14
      2.1.6 Delay-Path Analysis 14
      2.1.7 Design for Test 15
      2.1.8 Trojan Removal 16
   2.2 Testbed 16
      2.2.1 Structure 16
      2.2.2 Hardware 17
         2.2.2.1 Virtex 5 17
         2.2.2.2 Basys2 18
         2.2.2.3 MSO2014 18
         2.2.2.4 LA4850 18
      2.2.3 Software 19
         2.2.3.1 Xilinx ISE 19
         2.2.3.2 Synopsys WaveView 20
         2.2.3.3 Synopsys TetraMAX 20
         2.2.3.4 LabView SignalExpress 21
      2.2.4 Functionality 21
2.3 Dynamic Multi-Parameter Analysis
2.3.1 Dynamic Differential Functional Verification 23
2.3.2 Online Testing 24
2.3.3 High-Resolution Functional Profiling 25
2.3.4 Power Analysis 26
2.3.5 Supplemental Analysis Methods 26
2.4 Chapter Summary 27

3 Experimental Results
3.1 Setup
3.1.1 Controller Design 30
3.1.2 Test Pattern Generation 31
3.1.3 Testbed Assembly 31
3.2 Execution 32
3.3 Results
3.3.1 Design A Overview 33
3.3.2 Design A Trojan 1 and Trojan 2 34
3.3.3 Design A Trojan 3 and Trojan 4 34
3.3.4 Design A Trojan 5 and Trojan 6 35
3.3.5 Design B Overview 35
3.3.6 Design B Trojan 1 36
3.3.7 Design B Trojan 2 36
3.4 Technique Evaluation
3.4.1 Advantages 38
3.4.2 Disadvantages 38
3.5 Design B Clock Complications 39
3.6 Chapter Summary 41

4 Conclusions and Future Work
4.1 Conclusions 42
4.2 Future Work 42

References 44

Appendix A Oscilloscope Measurements 51
Appendix B Waveforms 55
Appendix C High-Resolution Output Waveforms 58
Appendix D Glossary 60
List of Tables

Table 1  Property Affected by Trojan  
Table 2  Design A Results  
Table 3  Design B Trojan Effects and Method of Detection  
Table 4  Design B Results  

iii
List of Figures

Figure 1  Hardware Trojan Taxonomy Proposed by Rajendran et al. 4
Figure 2  Block Diagram of Testbed 17
Figure 3  Representation of Controller Structure 22
Figure 4  Testing or Bypassing an Online System 24
Figure 5  Generic Control Unit Algorithm 29
Figure 6  Experimental Testbed 32
Figure 7  50MHz Clock Superimposed on 5MHz Clock 39
Figure 8  Clock Signal (Yellow) Behavior After Reset (Blue) 39
Figure 9  Clock Signal Subject to Possible Feedback Loop 40
Figure A-1  Scope Measurements for Design A, Trojan 5, 1.bit 51
Figure A-2  Scope Measurements for Design A, Trojan 5, 2.bit 51
Figure A-3  Scope Measurements for Design A, Trojan 5, 3.bit 52
Figure A-4  Scope Measurements for Design A, Trojan 5, 4.bit 52
Figure A-5  Scope Measurements for Design A, Trojan 6, 1.bit 53
Figure A-6  Scope Measurements for Design A, Trojan 6, 2.bit 53
Figure A-7  Scope Measurements for Design A, Trojan 6, 3.bit 54
Figure A-8  Scope Measurements for Design A, Trojan 6, 4.bit 54
Figure B-1  Output Values for Trojan A1 55
Figure B-2  Output Values for Trojan A2 55
Figure B-3  Test Overview for Trojan A3 55
Figure B-4  Output Values for Trojan A3 56
Figure B-5  Test Overview for Trojan A4 56
Figure B-6  Output Values for Trojan A4 56
Figure B-7  Power Measurements for Trojan A5 57
Figure B-8 Power Measurements for Trojan A6 57
Figure C-1 High-Resolution Output Capture for Design B, Trojan 1, 1.bit 58
Figure C-2 High-Resolution Output Capture for Design B, Trojan 1, 2.bit 58
Figure C-3 High-Resolution Output Capture for Design B, Trojan 1, 3.bit 59
Figure C-4 High-Resolution Output Capture for Design B, Trojan 1, 4.bit 59
Abstract

Hardware Trojan Horses (HTHs or Trojans) are malicious design modifications intended to cause the design to function incorrectly. Globalization of the IC development industry has created new opportunities for rogue agents to compromise a design in such a way. Offshore foundries cannot always be trusted, and the use of trusted foundries is not always practical or economical. There is a pressing need for a method to reliably detect these Trojans, to prevent compromised designs from being put into production.

This thesis proposes a multi-parameter analysis method that is capable of reliably detecting function-altering and performance-degrading Trojans in FPGA bitstreams. It is largely autonomous, able to perform functional verification and power analysis of a design with minimal user interaction. On-the-fly test vector generation and verification reduces the overhead of test creation by removing the need to pre-generate and verify test vector sets.

We implemented the method on a testbed constructed from COTS components, and tested it using a red-team/blue-team approach. The system was effective at detecting performance-degrading and function-altering embedded within combinational or sequential designs. The method was submitted for consideration in the 2012 Embedded Systems Challenge, which served to independently verify our results and evaluate the method; it was awarded first place in the competition.
Consider the following hypothetical situation: Alice is a fabless integrated circuit (IC) designer. She designs, builds, and tests IC designs, but does not have access to trusted fabrication facilities [1]. She sends her design to an off-shore foundry owned by Bob. Unfortunately, honesty is not one of Bob’s personality traits. During the fabrication process, he makes slight modifications to the design. These modifications (or “Trojans”) add malicious functionality to the design, causing it to exhibit undesired behavior. The Trojans that Bob inserted serve one of two functions: they cause a design to function incorrectly or they reduce the reliability of the design. To minimize the chances of his treachery being discovered, he only modifies some of the fabricated ICs.

Alice gets word of Bob’s betrayal and weighs her options. Contracting the work to another foundry for re-fabrication would be too costly and will take too much time. Alice decides to put the uncompromised designs into production and use a different foundry for the next batch. Except Bob wasn’t kind enough to tell her which ICs are compromised. She only knows the potential effects that the Trojan may exhibit: it either alters the functionality or increases power consumption. Alice needs to develop a method to test each of the chips and classify them as trusted or compromised.

This type of scenario is becoming all too common in the IC design industry. As the trend towards globalization increases, opportunities are created for rogue agents to modify the design during the fabrication process [2].

The above scenario was the basis for the 2012 Embedded Systems Challenge (ESC) organized during the Cyber Security Awareness Week (CSAW) event hosted by The Polytechnic Institute of New York University (NYU-Poly) [3]. Ten teams from three continents assumed Alice’s role of a betrayed designer. The teams were provided with the hardware description language (HDL) code of the original designs and a set of compiled field pro-
grammable gate array (FPGA) bitfiles\(^1\) that represent the chips returned from the foundry. Each team was given one month to classify as many of the files as possible as either compromised (containing a Trojan) or clean (not containing a Trojan) by creating a system which could test all of the designs and detect signs of malicious modifications.

The method proposed in this thesis was developed for and submitted to ESC 2012. It was evaluated for robustness, flexibility, complexity, and effectiveness by a panel of judges from industry and academia. Our technique was awarded first place based on our results and the characteristics of our hardware Trojan detection technique.

1.1 Thesis Organization

The remainder of this Thesis is organized as follows. Sections 1.2 - 1.4.3 describe the various classes of Trojans and their behaviors, and Section 1.5 provides a discussion of the differences between FPGA and IC designs, specifically how they affect the outcome of the tests. A comparison and discussion of various existing hardware Trojan detection techniques is provided in Section 2.1, along with a brief discussion of design strategies which can be used to increase the testability of the device.

Our detection method is described in Section 2.3, which covers details about the various phases of testing and the types of Trojans it is designed to detect. Section 2.2 discusses the physical implementation of the system and provides details about the various hardware components used. An overview of the experimental setup is provided in Section 3.1, and the results are reported in Section 3.3.

The overall effectiveness of the system is evaluated in Section 3.4 and the advantages/disadvantages are discussed. Finally, potential extensions and refinement to the system are presented along with our conclusions in Section 4.

The waveforms and figures supporting the results are provided in the Appendices. Oscilloscope power data is provided in Appendix A. Logic waveforms are presented in Appendix B. Functional profiling results are reported in Appendix C.

\(^1\)Binary representations of complete synthesized designs used to configure an FPGA device.
1.2 Hardware Trojan Horse

A hardware Trojan horse (HTH or Trojan) is a modification of an IC or FPGA design intended to change the design’s function. The Trojan is typically malicious in nature and is designed to have an undesirable effect on the target design.

Trojans are designed to operate without being detected. They have minimal overhead and negligible collateral impact on the functionality of the system. A well-designed Trojan can be nearly invisible to any analysis method. It can lie dormant within the design, completely inert until it receives some activation signal. This has led to a demand for more comprehensive detection methods that are able to reliably detect any undesired behavior in a design.

1.3 Trojan Taxonomies

Trojans are classified based on their physical characteristics, such as behavior and design. The first taxonomy was developed by Wang, Tehranipoor, and Plusquellic [4,5]. It uses the Trojan’s physical, activation, and action characteristics as classification parameters. Chakraborty presents an alternate taxonomy in [6] which classifies Trojans based on their trigger and payload characteristics.

Rajendran et al. have proposed [7, 8] the most comprehensive Trojan taxonomy method to date. It extends Tehranipoor’s work to use five characteristics for classification (Figure 1), covering a broader range of Trojans and offering a more detailed description. The taxonomy uses both design and implementation properties for classification, including the phase in which it was inserted, the abstraction level, activation requirements, the effects it has on the design, and the actual location of the Trojan within the design.

1.4 Trojan Design

A Trojan is designed to add malicious functionality to a design while remaining undetectable. It can have a wide range of effects on a number of different systems. A Trojan can remain idle, using virtually no power and having no effect on the system until it
is activated, or have such a subtle effect that it seems as though there is no effect. It must be custom built to target a specific design in order to be effective.

There are two basic Trojan design styles: combinational and sequential. Combinational Trojans are the easiest to test for, as they do not have any internal data and thus do not require an activation sequence. Sequential Trojans are more complex and often much more difficult to detect. They usually require an activation sequence before they exert any effect on the system.

1.4.1 Effects

The primary consideration in the design of the Trojan is the effect it is going to have on the target system. There is almost no limit as to what effects can be implemented.

A common effect is a functional alteration. At some point during operation, the system produces an incorrect or unexpected output or exhibits undesired behavior. This type of Trojan can be as simple as an added inverter or interconnect, or complex enough

Figure 1: Hardware Trojan Taxonomy Proposed by Rajendran et al. [7].
to cause a minor glitch in one specific case during operation. Regardless of the complexity, they still alter the functionality of the design.

Another common effect is reliability degradation of the design. Components are added to increase the power consumption of the design, which results in an increase in the temperature of the IC. The increased temperature causes accelerated degradation of the transistors, resulting in a shorter functional lifespan [9]. The same effect can be achieved by introducing hidden defects into the design during fabrication [10].

Some Trojans do not directly affect the target system at all. Instead, they are designed to leak or transmit information to a third party. This information can be operational data, encryption keys, or design details - any information that would not be available otherwise. There are various methods of sending this data. It may employ an antenna and transmission system to send the information via radio waves [11] or it may override the normal functionality of the chip to output the leaked data.

Lin et al. [12] proposes an interesting transmission technique which induces physical side-channels to convey information. The data is encoded into the power side-channel. It enables/disables embedded components to induce minute variations in the power signature. They were able to extract the data via power analysis. The process is very similar to a side-channel based watermarking technique proposed by Becker et al. [13].

1.4.2 Activation

The next consideration is the mechanism which will activate the Trojan. Some Trojans are active whenever the design is active, while others require a specific condition to occur before it activates. This is useful for hiding the Trojan and masking its functionality. Since the Trojan has to be activated before it has any effect, it can be very hard to detect with traditional detection methods.

Some activation requirements are relatively simple, simply requiring a specific sequence of input combinations. For very large or complex designs, a multi-value activation sequence can exponentially increase the complexity of the test. Some methods exist, as will be discussed later in this section, which can attempt to reduce the test complexity.
Sophisticated Trojans targeting highly complex systems can also have highly complex activation mechanisms. Gao et al. [14] demonstrate that a Trojan can be embedded into a processor and controlled via code. The method was independently verified by a University of Illinois team in [15]. These Trojans have higher destructive potential and are much more difficult to detect than non code-controlled Trojans. They can provide direct access to software stack and processor components.

Code-controlled Trojans can be activated by a specific sequence of commands or events. Once activated, they can inject code or data into the processor, modify the functionality, or leak information. Alternatively, they can be designed to activate when specific values are stored in certain registers. The potential complexity and uniqueness of the activation sequence makes these Trojans very difficult to test for.

1.4.3 Implementation

The final consideration is how exactly the Trojan is to be implemented into the target system. Most simple Trojans are just inserted into the appropriate components.

Alkabani and Koushanfar developed a method [16] of inserting a Trojan before the target design is synthesized. The Trojan gives the attacker low-level control over the design, and has very low overhead, partially because the synthesis algorithm attempts to optimize the design at runtime. The process optimizes the entire design, including the Trojan, minimizing power consumption and area. This makes the Trojan more difficult to detect as it is fundamentally optimized into the target design.

If a Trojan cannot be inserted before synthesis, it can be directly integrated within a design to minimize its footprint. A method is proposed by Rahmatian et al. [17] in which the Trojan is implemented using existing gates in the target system. This reduces the amount of circuitry added to the design, minimizing its impact on area and delay.

1.5 FPGA Considerations

By design, FPGA systems have fundamentally different operation than traditional complementary metal-oxide semiconductor (CMOS) application-specific integrated circuit (ASIC) designs. Specifically, power analysis must be performed in a slightly different man-
ner than with an ASIC. Since the FPGA uses look-up tables (LUTs) and registers for design storage and operation [18], the power signatures will be vastly different than for a functionally identical ASIC design. It is shown in [19] that much of the power consumption is by the routing and clocking resources, characteristics that CMOS ASIC designs typically do not share. Additionally, the signatures will be different for the same design on different FPGA devices, due to differences in design and construction.

Design of our power analysis procedure was derived from the method used in [20] to perform power analysis on FPGA-based cryptographic systems. FPGA power analysis techniques were initially researched as methods of reverse-engineering secure designs. Due to their flexibility, FPGAs offer many advantages over traditional ASIC designs. As discussed, FPGAs and ASICs have fundamentally different power signatures, but it is shown in [21] and [22] that power analysis is equally effective and accurate when testing either an FPGA or an ASIC design.

FPGA-based designs offer some distinct advantages, from a testing perspective. The ability to augment or modify the design on-the-fly enables advanced detection techniques, such as the insertion of diagnostic components. Additionally, a single FPGA design can be implemented on multiple pieces of silicon, mitigating the effects of process variation by normalizing the measurements and results. Maggioni [23] presents an FPGA-based system capable of self-diagnosis and verification, through the use of on-board trust verification subsystems.

CSAW 2008 included another embedded systems challenge. The objective was to implement multiple Trojans into a secured FPGA design. The HDL source was not provided for analysis; the teams had to reverse-engineer the design themselves [24]. The first-place team was able to implement a vast range of Trojans with a very low degree of detectability [25]. Their Trojans were designed to leak information, and had virtually no overhead that could be detected. A second team [26] was also able to successfully embed a wide range of Trojans within the design that were very hard to detect. A third team [27] was only able to develop a small number of Trojans, but were successful in making them difficult to detect. These served to prove that FPGAs are extremely vulnerable for hosting hard-to-detect Trojans.
1.6 Chapter Summary

Hardware Trojans are a real and present security concern for hardware designers. There are many methods of designing, constructing, and implementing a Trojan. A comprehensive method must be developed to verify that a design has not been tampered with.
2 Proposed Dynamic Multi-Parameter Analysis Technique

Detection methods have to be flexible and robust, since a hardware Trojan can be hidden anywhere in the design, such as within a processor [15] or embedded in a third-party black-box intellectual property (IP) core [28]. It needs to be able to detect a variety of Trojan designs, with minimal false positives or false negatives.

Many analysis techniques can be used for Trojan detection. These techniques are designed to test a specific property or function of a design. Many of these methods were developed either to verify design functionality after fabrication, or to exploit or attack a system.

The remainder of this chapter is organized as follows. Section 2.1 provides a survey of existing Trojan detection techniques. Our testbed design details are provided in Section 2.2, and the proposed method is described in Section 2.3. A summary of the chapter is provided in Section 2.4.

2.1 Hardware Trojan Detection Techniques - A Survey

Dozens of individual Trojan detection methods have been proposed, most of which are simple modifications of existing techniques. There are only a handful of distinctly different basic techniques in use, each of which has its own strengths and weaknesses. This section discusses common techniques currently in use, and evaluates their usability in our system.

2.1.1 Functional Verification

*Combinational Designs* One defining characteristic is that the output is a pure function of the current input only; the output does not rely on previous values or states. Any given input value will always produce a specific output response [29]. This property simplifies the verification process.
Given a combinational design with \( n \) inputs, the maximum number of input/output combinations that need to be tested to ensure full coverage is \( 2^n \). In some cases, a design may be small enough to justify the use of an exhaustive test set. For large designs, the runtime of the verification algorithm may be unacceptably high - a linear increase in the number of input bits results in an exponential increase in runtime [30].

Methods exist which may be able to reduce the number of test vectors required for full activation coverage. Various automatic test pattern generation (ATPG) algorithms can be used to create a range of full-coverage sets. However, reduced test sets are not guaranteed to provide any benefit, and in certain cases may prevent Trojan detection [4].

One reduction approach is to create a set that provides full fault coverage to activate as many nets as possible within the design [31], increasing the likelihood of Trojan activation. Another option is to generate a highly specialized set to increase the chances of Trojan activation, as discussed in [32]. Multiple reduced sets can be used together to increase the chances of activating the Trojan.

If the Trojan requires an activation sequence, the test vector set may be exponentially larger. A design with \( n \) inputs that contains a Trojan with an activation sequence of \( m \) values\(^2\) will have up to \((2^n)^m\) input/output (I/O) combinations. However, such a Trojan needs a subsystem in place to handle sequence checking and Trojan activation. This subsystem will increase the overhead of the Trojan, making it more visible to side-channel analysis.

**Sequential Designs** Sequential designs are more complicated than combinational designs. The output of a sequential design relies on both the current input and the previous inputs. As a result, full coverage functional testing is highly complex, and exhaustive testing is entirely impractical.

After a design has been fabricated, it is typically tested to verify correct operation. This requires functional testing with full or nearly full coverage, similar to the requirements for Trojan detection. To overcome some of the complexity, hardware designers usually insert components into the design to add test features.

\(^2\)Assuming that values may appear in the sequence multiple times.
A common practice is to include a scan-chain mode in the design to enable the loading of values into the internal registers. This effectively allows the user to set the state of the machine without requiring all of the prior inputs be applied. Similar to the combinational designs, an ATPG algorithm can be used to create a full-coverage test set.

Alternatively, as discussed in [33], a design may include power gating or multiplexing components to allow functional verification of individual components of the design, while keeping the rest disabled. This can reduce the complexity of the test, since it does not have to provide full coverage for the entire design at once. This is especially useful when testing system-on-chips (SoCs), as their size and complexity makes full-coverage testing difficult [34].

2.1.2 Side-Channel Analysis

Side-channel analysis (SCA) is the process of gleaning details about a design by inspecting its physical implementation properties. It is possible to determine the general composition of a design by measuring its physical characteristics. SCA techniques are commonly used in cryptanalysis to glean secure information, such as encryption keys, from a design. This is done by constructing a model from the collected data and analyzing it to extract design details.

SCA was first shown to be effective for Trojan detection by Agrawal et al. in [35]. It employs a destructive method on a small number of ICs to generate the initial model, and then performs single-point transient analysis to verify the rest of the designs. The experiment verified the usability of SCA for Trojan detection.

2.1.3 Power Analysis

Power analysis is the process of measuring various characteristics of the design related to power consumption and distribution. The current flowing through the voltage supply line ($V_{DD}$) is monitored and analyzed to detect any unusual or unexpected fluctuations. This data is typically collected using an oscilloscope using one of two methods: direct current measurement or differential voltage measurement. Direct current measurement requires that the oscilloscope be equipped with a current probe to directly calculate the power
consumption of the device. Direct measurement is the preferred method of collecting data, though a differential voltage measurement across a small resistor has been shown to work with an FPGA [20]. The differential voltage measurement method may be slightly less accurate than a current probe, however this can be compensated for during data analysis.

Power analysis is common in side-channel cryptanalysis, where it is used to sniff out encryption keys from power signatures. The technique was adapted for use in detecting Trojans [35]. It creates a set of fingerprints for a design through destructive analysis of a few samples of the chip. The signatures for the rest of the ICs were verified against this.

Differential voltage measurement uses Ohm’s Law (1) to calculate the current ($I$) given the voltage differential ($\Delta V$) across a known resistance ($R$). Solving (1) for $I$ results in (2) where the resistance and measured voltages can be substituted to calculate the current.

$$\Delta V = IR$$  

$$\frac{V_{source} - V_{load}}{R} = I$$

Power consumption is monitored and recorded during operation. The analysis can either be applied to the design as a whole or to a subsystem of the design. Localized analysis requires a more complex setup and the design being tested must be able to support this type of analysis. Implementations of this technique are featured in [36] and [37].

The process is subject to noise generated by the onboard components which may mask the signature of the Trojan design. The use of a multi-point power analysis, rather than single point source analysis, can enhance the effectiveness and sensitivity of power analysis, as discussed in [38] by Rad, Plusquellic, and Tehranipoor. They further enhanced the efficiency by calibrating the power supply signals to minimize noise [39].

A comprehensive model of the power consumption of the design is required for comparison. This can be a computer simulation model or a set of characteristic measurements from analysis of known good designs. This allows for statistical analysis of the results to be performed, and any verifiable outliers may be indicative of a Trojan.
Alternatively, some types of Trojan designs may be detectable without a baseline model to reference. If the Trojan has a delayed or triggered activation sequence, then it may be possible to measure a difference in power consumption after it has been activated.

Process variation can have a significant effect on the results of side channel analysis. A method is proposed in [40] and expanded upon in [41] which uses a self-referencing analysis technique to mitigate this effect. It entails comparing the power signature of one region of an IC with that of a separate region, enabling them to determine if an inconsistency in the signature is the result of process variation.

The authors of [42] propose an alternative method to overcome the process variation problem, which uses multi-parameter SCA and carefully designed test vectors to enhance the results of analysis. The vectors are designed to minimize component switching while maximizing Trojan activation potential.

2.1.4 Thermal Profiling

Thermal profiling is a theoretical technique that does not seem to have been studied much for hardware trojan detection. One method has been proposed which uses thermal conditioning with gate-level characterization (GLC) [43]. They apply different temperatures to specific separate areas of the chip and monitor the change in the characteristics of the gate. This enables them to perform GLC with a thermal conditioning method, but does not directly profile a design.

Direct thermal profiling would involve taking real-time high-resolution measurements of hotspots on the chip itself\(^3\). Depending on the resolution of the thermal imaging device, it may be possible to isolate specific components or areas of the chip to test. Lower resolution devices would only be able to analyze larger sections of the chip, making it more difficult to detect a small, cool Trojan design. As CMOS channel lengths continue to shrink and components become smaller, this method may be rendered impractical, if it is not already.

\(^3\)At the time of writing, a survey of available literature did not indicate that this avenue of testing is currently being researched or pursued.
2.1.5 Gate-Level Characterization

GLC is the process of creating a mathematical model of a system from data collected by side-channel and functional analysis methods. The model represents the actual implemented hardware, rather than the initial design. The authors of [44] propose a method of using this model to determine if a design has been modified. It uses GLC to create a system of linear equations, which is then solved to determine the makeup of the design. The model is then analyzed for inconsistencies with the original design to detect any sign of a Trojan.

Another application of GLC, described in [45], uses a GLC model of the circuit to analyze sampled side-channel data from the design. The side-channel data can be used to determine if a Trojan has been introduced into the system. This method is an extension of the work done in [46] which uses a consistency-based approach, rather than GLC modeling. Both methods use partitioning and segmentation to enable independent testing of regions of the design.

2.1.6 Delay-Path Analysis

Delay path analysis the process of determining the delay characteristics of signals in the design and verifying them against the GLC model. One method of delay path analysis involves modeling delay path elements in a combinational circuit to create a ‘fingerprint’ of the design [11]. Another approach is proposed in [47] which uses a similar method to model synchronous designs at operating speed. Comprehensive models of all internal components must be available for use in the model. If any component cannot be accurately modeled, the delay-path model will be incorrect.

Delay path analysis has been shown to be very robust detection method [48]. Of course, like all other methods covered so far, it is not 100% effective. This type of analysis also has the potential to have very high overhead, as the process of accurately modeling delay paths can be computationally complex. Further adding to the complexity is the need to compensate for any process variation when determining the delay.
2.1.7 Design for Test

All of these methods are limited by the testability of the design. In some cases, the test is unable to activate the Trojan, and thus is not able to observe its effects.

Salmani, Tehranipoor, and Plusquellic propose a method in [49] to enhance testability of a design for hardware Trojans. Ancillary flip-flops are strategically inserted into critical areas of the design. They are used during testing to minimize the time required for Trojan activation by activating internal nets that may not be normally activated. They expanded on their method in [50] by increasing the efficiency of the insertion process.

Chakraborty, Paul, and Bhunia propose in [51] a similar design-for-trust technique termed ‘on-demand transparency.’ A system is designed to operate in two modes: operational and transparent. In ‘transparent mode’ it can function similar to Salmani’s flip-flop insertion technique in that it induces rare events within the system to increase the probability of Trojan activation.

An alternative method involves using sustained test vectors to minimize internal flip-flop toggles [52]. These vectors, like those described above, are designed to maximize Trojan activation potential. This reduces the noise in the power signature caused by internal state changes.

Zhang and Tehranipoor propose a method [53] in which a ring oscillator network is distributed throughout the design. This can be activated during testing to mitigate measurement noise and provide localized analysis capabilities. The method is generally effective, decreasing the time to detect a Trojan. However, as shown in [54], a Trojan can be designed that is not affected by the network.

In some cases, it may be beneficial to prevent a Trojan from operating properly within the design. An interesting approach is to obfuscate the internal interconnect network so that a Trojan cannot obtain useful information. It also changes the internal functionality of the device, which may cause the Trojan’s triggering mechanism to fail.

Two separate methods are outlined in [55] and [56]. The former uses a 4-step process to identify and obfuscate critical signals, while the latter proposes an automated methodology to obfuscate the behavior of the design.
2.1.8 Trojan Removal

Many detection and prevention methods have been proposed and proven to be effective at detection, but there is one issue which is rarely addressed. Once a Trojan has been successfully detected in a system, the options for followup are extremely limited. Due to the nature of CMOS designs, a Trojan cannot simply be removed from the system. Once it has been embedded, it is there to stay. Soft designs such as FPGAs can be rapidly rebuilt with minimal overhead after the Trojan has been isolated and removed. CMOS designs must be discarded and new chips have to be fabricated. This is obviously less than ideal.

A method proposed by Wei and Potkonjak in [46] seems to be an ultimate solution to the problem. Their design uses an accelerated aging mechanism to physically disable the Trojan, similar to the mechanism used by some Trojan designs. The aging effect is highly localized to minimize collateral damage. This method provides the potential to recover a compromised design without requiring re-fabrication. Implementation of this method requires that the system be designed for it, which may not be feasible in most situations.

2.2 Testbed

The experimental testing setup (or testbed) is designed for speed and flexibility. The system consists of two FPGA devices, a mixed-signal oscilloscope, and a high-bandwidth logic analyzer, as shown in Figure 2. It is designed to execute all test phases and automate data collection.

2.2.1 Structure

The main component of the testbed is the Virtex 5 FPGA device. The control unit, golden model, and ATPG subsystem(s) are all contained on the device. The unit under test (UUT) is programmed onto the Basys2 FPGA. A 16-bit I/O bus connects the two devices together. Taps are inserted into the I/O bus to relay the data to the logic analyzer and the oscilloscope’s digital input lines.

Power for the Basys2 is provided by the Virtex 5. It delivers a steady, low-noise 5V supply suitable for use with power analysis. Two leads connect the boards, one for $V_{DD}$
and one for \textit{gnd}. A 10\,\Omega resistor is inserted in the $V_{DD}$ lead for use with differential voltage measurements. Voltage probes from the oscilloscope are connected on either side of the resistor for differential voltage measurements.

2.2.2 Hardware

Most of the hardware components were selected due to availability; equipment is limited due to budget and resource constraints. The Embedded Systems Challenge required that a Digilent Basys2 FPGA be used as the carrier for the UUT.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{testbed.png}
\caption{Block Diagram of Testbed}
\end{figure}

2.2.2.1 Virtex 5

The control unit and golden model are both configured on a Xilinx Virtex 5 FPGA alongside the test pattern generation/storage module(s). It provides a large number (16 differential input pairs, 32 single-ended inputs) of I/O ports, high-precision clocking, and a large number of slices to hold design information. Additionally, it provides access to regulated +5V DC power supply pins, which are used to provide power to the Basys2.
2.2.2.2 Basys2

As required for the competition, the UUT was configured onto a Basys2 FPGA board [57]. The device offers very little in the way of functionality; it has limited I/O availability, low operating frequency, and limited clocking capabilities. One benefit is that it requires little power, allowing it to be easily powered by the Virtex 5. The simplicity also results in a cleaner power signature, since there isn’t extra hardware onboard that may affect the signature. For most of our needs, it performs well enough.

2.2.2.3 MSO2014

The oscilloscope used in the system is a Tektronix MSO2014 Mixed-Signal Oscilloscope. The device offers 4 analog channels for voltage measurements and 16 digital channels, and has up to 200MHz bandwidth at 1GS/s [58, 59]. It has the ability to store collected data on a USB flash drive for export to a PC as either a comma-separated value (CSV) formatted data file containing all data points or a screenshot image of the currently display.

Advanced analysis capabilities are available for the analog channels. For example, it can perform simple mathematical operations on waveforms in real-time. We use an invert-and-add function derived from Equation 1 to calculate the power use from the differential voltage measurements.

Additional features can be activated with the use of an application module. For this setup, the Embedded Serial Triggering and Analysis module was used. This module enables advanced functionality for working with embedded systems [60]. Specifically, it allows multiple digital input signals grouped into a bus to be used as a triggering mechanism for the scope. It can trigger on a specific input value or sequence, enabling the construction of complex trigger conditions for specific scenarios.

2.2.2.4 LA4850

A Link Instruments LA4850 80-channel logic analyzer was used for collecting I/O data during testing. It can operate at much higher frequencies than the FPGA device,
maxing out at 500MHz. Each input channel can buffer 512K samples which can be read with bundled software [61].

The device is modular, utilizing ‘pods’ that are connected to provide the desired functionality. There are two types of pods, logic pods (inputs) and pattern generators (outputs). Each pod provides 8 channels, and the device can be equipped with up to 10 pods. The pattern generator pods can be used to apply test patterns in cases where the Virtex 5 is not sufficient.

Due to its status as a discontinued product, the device is no longer supported by the vendor. The provided drivers are incompatible with modern operating systems. This limited the usefulness of the device for advanced control features. An obvious remedy to the situation is to upgrade to a supported device.

2.2.3 Software

A number of software applications are used to augment and automate various phases of the testing procedure. Following are the primary software tools we use for design and testing. This section provides a brief description of the tools we use in testing.

2.2.3.1 Xilinx ISE

The entirety of the control unit design and testing was done with the Xilinx ISE Design Suite, an integrated design environment (IDE) for creating, testing, and deploying FPGA designs [62]. It handles the entire workflow from Verilog to Bitfile generation, automating synthesis, place-and-route, translate, and compilation.

An integrated testing environment is provided through ISim, the associated HDL simulator [63,64]. Both pre- and post-synthesis models can be simulated for function and performance. This ability can also be used as a preliminary test to detect Trojans inserted by the synthesis tool.

ChipScope is a comprehensive on-board diagnostic system for use in Xilinx FPGA devices [65]. It consists of a set of IP cores and a GUI. The cores are generated from within ISE and instantiated in the design. There are two available diagnostic modules, one which
can only read signals, and one which can function as a limited I/O module. The data is accessed by the software via a JTAG programming cable.

PlanAhead is a design planning tool used to optimize placement of pins and components on the FPGA. It was used to determine the optimal pin usage for the control unit inputs and outputs. The latency and buffer information for each pin is provided, and floor-planning analysis can be performed on a pre-synthesis or post-synthesis design. Advanced performance analysis of the control unit provided information about the limitations of the design.

2.2.3.2 Synopsys WaveView

A waveform viewer can be used to perform additional analysis on the data collected by the oscilloscope. It is typically more comprehensive and easier to use than the scope itself. We used Synopsys WaveView, which provides extensive analysis and post-processing capabilities [66]. The program is designed for use with netlisted designs and SPICE simulations, and is not quite optimal for this application. This was not a serious limitation in our case, since we primarily used it to display information rather than for analysis.

The data has to be available locally for the program to access it. This required us to manually transfer the data from the oscilloscope to a computer, a process which ultimately proved impractical. The scope’s export ability was limited, extremely slow, tedious, and prone to errors. We finally decided to just use the waveform images generated by the oscilloscope for data collection. These waveforms are provided in Appendix A.

2.2.3.3 Synopsys TetraMAX

As discussed previously, the control unit is able to use pre-generated test vectors as input stimuli. We used Synopsys TetraMAX, to analyze the HDL design and create a targeted test vector set. TetraMAX is an automated test pattern generator capable of analyzing complex sequential or combinational designs and generating full-coverage test sets. Parameters and specifications are provided by the designer, and TetraMAX generates tests for the specific scenario [67].
ATPG software such as this can potentially be used to accelerate functional verification by reducing the number of test vectors. The reduced sets must be carefully designed to provide maximum coverage. These sets may not be effective for Trojans that require an activation sequence, since some of the values required for the sequence may not be present in the reduced set.

2.2.3.4 LabView SignalExpress

Many modern oscilloscopes provide an interface to communicate with a PC, usually via the interchangeable virtual instrument (IVI) protocol. Software on the PC can send commands to and collect data from the oscilloscope. Collected data is stored on the PC and manipulated/analyzed by software. The MSO2014 provides such an interface, though it is very limited [68] and is not fully IVI compliant.

SignalExpress Tektronix Edition from National Instruments is the PC software side of the scope interface. It can relay and analyze real-time data from the scope as well as control most of its functionality. The IVI compatibility issue prevents the software from accessing digital signal information, advanced trigger controls, mathematical transformation functions, or the triggering features provided by the Embedded System module. It can retrieve data from the four analog channels and control very basic triggering data, but that is the extend of its abilities. Due to the limitations, the software ultimately proved to be impractical to use and provided no benefit.

2.2.4 Functionality

The testbed is divided into three functional sections: the control unit, the UUT, and the data acquisition devices. The control unit is contained on the Virtex 5 FPGA, and the UUT is implemented on the Basys2. The oscilloscope and logic analyzer acquire side-channel and functional data during testing.

The controller, written in Verilog, consists of a finite state machine (FSM), I/O modules, and a HDL representation of the design being tested. A block-level representation of the controller’s structure is shown in Figure 3. The FSM controls the entire test procedure. This procedure is different for each design, but consists of the same basic steps.
First, it either generates test vectors or reads pre-generated vectors from a read-only memory (ROM). It then applies these vectors to both the design being tested and the golden design. The output response of the UUT is verified against the Golden Model output. In the event of a failure, it signals the user and halts the test.

Once it has tested all necessary input vectors, it starts again from the beginning. This cyclical testing enables further data collection for side-channel analysis. The control unit outputs a synchronization signal for the oscilloscope and logic analyzer, providing a constant reference point.

The oscilloscope averages the measurements taken over 512 iterations, mitigating much of the noise in the power signature. After 512 iterations of the test application sequence, the test is concluded and analysis of the side-channel data begins.

2.3 Dynamic Multi-Parameter Analysis

The detection method proposed in this thesis incorporates a number of the methods described in Section 2.1 to provide a comprehensive detection package. Due to the variety of hardware Trojans and the range of effects they can have, a single detection method is unlikely to be robust enough for universal detection.
2.3.1 Dynamic Differential Functional Verification

The proposed system uses dynamic differential functional verification (DDFV), a functional verification method that can be adapted in real-time to respond to changing test conditions. The configuration and test parameters can be reconfigured without requiring a complete re-implementation. It can be configured to detect a wide range of Trojans.

The core of the algorithm is a generic functional verification algorithm. It selects a test vector, applies it to the system, and validates the output response. If the output is incorrect, the algorithm halts in an error state. This base functionality is extended to provide dynamic testing capabilities such as dynamic vector generation.

The test vectors are typically pre-generated by an ATPG algorithm, and stored in memory onboard the testing device. Partial runtime reconfiguration of the FPGA [69] can enable augmentation or modification of the stored vectors while the device is online. Alternatively, the controller can be configured with an onboard ATPG module to generate the vectors in realtime. A simple example of this is a counter, to provide an exhaustive set of vectors. A third option is to read vectors from an external source, though extensive modification may need to be done to the algorithm to keep it in sync with the input data stream.

Once all the test vectors have been verified, the algorithm can either stop and indicate success or reset and repeat the test. This gives multiple opportunities to activate delay- or sequence-triggered Trojans. It also provides the ability to perform consistent, long-term power analysis.

Unlike other functional verification methods, this algorithm does not rely on pre-generated outputs, simulation results, or mathematical modeling. It uses a differential verification method to generate outputs on-the-fly. A verified, accurate HDL model of the original design\(^4\) (the ‘Golden Model’) is instantiated alongside the control unit. The algorithm applies the test vector to the UUT and the Golden Model simultaneously, and compares the resulting output responses.

\(^4\)Golden model must be verified as clean. If the Trojan was inserted during the design phase and is present in the model, detection will likely fail.
2.3.2 Online Testing

The algorithm can be adapted to test a device that is online in the field. This method is more complex than the other functional verification methods. It involves testing a system that is active and functioning in a production environment. The device is actively receiving data from external systems, and its output is being sent to other external systems. The trick is in the ability to run the test without interrupting the operational data stream.

The control unit can be configured to use the operational input to the UUT as the test vector source, as shown in Figure 4(a). Rather than output the vector to both designs, it only sends it to the Golden Model. The controller monitors the UUT output data and compares it with the Golden Model output. If the UUT deviates from the expected output, it may indicate the presence of a Trojan. The controller can store data during the test which can be exported and analyzed to determine the cause.

This approach may not be applicable to some situations, especially when testing sequential designs. Combinational designs have a constant input-output mapping [29], so the controller will not require any major modification in order to function. The Golden Model has to be able to match the delay characteristics of the UUT to provide accurate results, or the outputs from the UUT and the Golden Model will not be synchronized.

![Figure 4: Testing (a) or Bypassing (b) an Online System (Green lines represent Golden Model output, orange lines indicate UUT output)](image)

Figure 4: Testing (a) or Bypassing (b) an Online System (Green lines represent Golden Model output, orange lines indicate UUT output)
Sequential circuits are different from combinational circuits in that their output relies on both the current input and the current state. As a result, the online testing process requires a more complex control unit. The controller will need to be able to synchronize the states before initiating the tests, and it must be able to do it without interfering with the normal operation of the UUT or the data flow. The process is limited by maximum operating frequency of the FPGA; if the UUT operates at a higher frequency than the FPGA device, online testing will not be possible.

This method of online testing has a hidden benefit. If the UUT is found to contain a Trojan, the controller and Golden Model can be used to bypass the device until it can be replaced. This can be achieved by re-configuring the inputs and outputs to route around the compromised device (Figure 4(b)), allowing for its removal. The Golden Model output replaces the UUT output.

2.3.3 High-Resolution Functional Profiling

Virtex 5 is not an ideal platform for a high-frequency logical analysis due to its relatively low operational frequency. The LUTs on the Virtex 5 are driven by a 200MHz onboard clock signal [70], and the process of reading and storing an input takes multiple5 clock cycles. As a result, the input sampling rate of the design may be too low to reliably detect brief glitches in the output.

To address this limitation, a commercial logic analyzer is attached in parallel with the Virtex 5 which collects I/O data from the UUT at a very high sample rate - much higher than is possible with the Virtex 5. This data is used to create a functional profile of the design that describes its overall behavior. The profile is analyzed to determine if the behavior of the design is consistent with the expected behavior. The collected data is represented as a waveform (see Appendix C) which is manually verified through a visual inspection6.

5The exact number depends on the design of the control unit, specifically the FSM which enables input capture.
6Visual inspection may not be practical in most cases. An alternative is to use statistics software such as Minitab to analyze the raw data. Computational environments such as Matlab or Mathematica can be used to automate data processing tasks.
The large amount of data collected provides us with additional analysis options. Waveforms taken at different points in the testing process can be compared to detect any subtle changes in delay or signal skew over time. We are able to observe glitches or output signal instabilities that are too brief to be detected with functional verification. These measurements are not possible with the Virtex 5 controller alone due to the mentioned limitations.

2.3.4 Power Analysis

Power analysis uses a mixed-signal oscilloscope with analog and digital signal capture capabilities. For our system, we only needed to measure two voltage points, requiring two analog inputs; advanced triggering used 8 digital inputs.

Baseline measurements are taken from trusted devices to use for comparison. Power consumption of the UUT is monitored for the duration of the test. Consistency analysis [71] can be used to detect any inconsistencies which may indicate a Trojan. The currently implemented method uses the magnitude of average power consumption over the whole test, meaning that it can only detect Trojans that cause a noticeable increase in power consumption. This proved sufficient for our purposes.

2.3.5 Supplemental Analysis Methods

In the event that a Trojan cannot be physically detected, there may be a supplemental method available. This is technically a form of SCA, as it analyzes the properties of ‘soft’ digital representation of the designs to verify trust. The usefulness of the method is limited by the availability of soft designs existing for both the clean and compromised designs.

FPGA-based designs have one key characteristic which sets them apart from custom ASIC designs - an FPGA design is represented as a binary bitstream which is used to program the device. This bitstream is a direct representation of the design, and each bitstream is unique for the design it represents. This provides for more testable parameters for use in classifying FPGA-based designs; we can prove that two designs are identical if we can demonstrate that their representative bitstreams are identical.
ASIC designs are not volatile, and as such have no corresponding bitfiles for programming. This limits the possibility of a soft analysis method. Digital representations of ASIC designs typically only exist during the design process or when being transported to the fabrication facility.

Such a method can be used to verify the design files during the design process, to ensure that no files have been modified without authorization. After the design has been transferred to the fabrication facility, this method becomes irrelevant; if the fabrication facility cannot be trusted, it is unlikely that a facility will provide the modified files of an intentionally compromised design. Given these limitations, it is unlikely that a soft verification method can be used for ASIC designs since the source materials may not be trusted.

2.4 Chapter Summary

Many methods exist to detect Trojans. Each is designed for use in detecting a specific type. Our method uses two common techniques, power analysis, and functional verification, to detect some common Trojan types. The testbed is designed to allow for rapid testing and consistent data collection, and is used to evaluate the proposed method.
3 Experimental Results

The parameters of the experiment were defined by the judges of the Embedded Systems Challenge. We were given a series of tasks to complete, and were judged on the effectiveness and flexibility of the proposed detection method. The test cases represented a range of common Trojans with different effects and activation mechanisms, embedded in both sequential and combinational designs.

The competition was set up using a red-team/blue-team approach. The red team represents a rogue agent with malicious intent. Their objective was to create multiple variations of a target design, with each variation containing a unique Trojan with its own effect and activation requirements. Each Trojan exhibited at least one of two general effects; it could (i) cause an incorrect output value to be produced for one or more input values, and/or (ii) increase the power consumption of the design by at least 5%\textsuperscript{7}.

A blue team represented a loyal verification team. Their responsibility was to develop and implement a method of detecting Trojans, and to determine which designs have been compromised. To be successful, the team must correctly identify all of the compromised designs. The teams’ methods were evaluated by a panel of judges. The evaluation consisted of quantitative as well as qualitative assessment. The quantitative assessment was based on the performance of the test system with respect to the competition only. The qualitative assessment analyzed the general abilities of the method, such as scalability, performance, complexity, and flexibility.

3.1 Setup

Our method requires that the test be custom designed for use in a particular situation. It is impractical to create a large monolithic design capable of testing any design of any size for any possible Trojan. Instead, we build a few specialized tests that are very fast

\textsuperscript{7}For this competition, the designs have been constructed to mitigate the effects of process variation.
and efficient. This design style also makes it easier to integrate the golden model and test generation subsystems, since the controller will be specifically designed to work with them.

Initially, we considered creating a control unit capable of executing the tests for both designs. This turned out to be impractical, as the test procedure for a combinational design is fundamentally different than that for a sequential design. A unified control unit would also be more complex and less efficient. We decided to create two separate control units. Both of the units will perform the same basic tasks: (i) retrieve test vector from ATPG or ROM, (ii) apply test vector to inputs of UUT and golden model, and (iii) verify the output response of the UUT against the golden model. The value is held for 160ns before the output is verified, to allow for signal stabilization. The algorithm is illustrated in Figure 5. The actual implementations are more complex, but the premise is the same.

Functionally, the test procedure for the combinational design isn’t much more complex than the generic algorithm. We added a continuous testing ability that restarts the test at the end of each iteration. This can be used to detect Trojans with simple activation sequences or to perform power analysis for a sustained period of time. If any test fails at any point, the program halts and signals an error condition to the user. The input vector that resulted in a failure is displayed in binary on indicator light-emitting diodes (LEDs). Chipscope modules embedded within the control unit can be used to view the applied input value and the output values from the UUT and golden model.

The test procedure for the sequential design is more complicated, since the output is dependent on the previous states of the design. Exhaustive testing is simply not an answer. The design contains scan-chain access to the internal registers, which allowed us

---

**Data:** A list of input test vectors `tvecs`

```plaintext
foreach vector v in tvecs do
    UUT.setInput(v);
    goldenModel.setInput(v);
    sleep(160ns);
    if UUT.out ≠ goldenModel.out then
        return Fail;
return Pass;
```

Figure 5: Generic Control Unit Algorithm
to set state information without having to progress through all the states. Scan-in and scan-out can only be performed while the UUT is in test mode, and I/O is only active in normal operation. Input and output phases are each split into two stages to accommodate this behavior. The input phase performs scan-in before applying the input patterns and disabling test mode. Test mode is only disabled for one clock cycle. When it is re-enabled, the output is collected and verified by the control unit. Data from internal registers is retrieved through scan-out and is also verified against the golden model.

This process requires that the UUT and the golden model are completely in sync. Both should get the same values at the same time, and their triggering clock signals have to be in phase with each other. Otherwise, data may not be transmitted correctly to one of the designs, resulting in a test failure. The UUT in this case utilized a 5MHz clock signal ($clk_{ref}$) that could be referenced via an output pin on the Basys2. We designed a module to generate and synchronize a usable clock signal. The module samples $clk_{ref}$ on the positive and negative edges of the control unit’s 100MHz clock. In theory, this module is capable of generating a clock with the same frequency as $clk_{ref}$ that is no more than 5ns out of phase. In practice, hardware limitations prevented the module from functioning at all. A further discussion and analysis is provided in Section 3.5.

3.1.1 Controller Design

The experiment called for testing two separate designs, referred to as Design A and Design B. Design A is a combinational circuit, while Design B is a sequential circuit. Since the two designs are so different, a separate control unit was constructed for each.

The Design A controller is a simple design, since functional verification of combinational designs is a relatively straightforward process. It can read pre-generated test vectors from onboard ROM or generate a test set using the onboard ATPG module. The ATPG module consisted of a single 8-bit counter that is incremented if the UUT passes verification with the current test vector. This module is specifically constructed to produce an exhaustive test set, and takes advantage of the relatively low number of possible input values. More complex designs will likely need a more advanced ATPG algorithm to produce a reduced test set, since exhaustive testing may result in an unacceptably long runtime.
The controller for Design B is decidedly more complex due to the fact that Design B itself is far more complex. The controller is only designed to read pre-generated vectors from a ROM; an ATPG module is not included, and exhaustive testing was not considered. The test vectors consist of a 145-bit scan vector and a 5-bit input vector.

3.1.2 Test Pattern Generation

The control unit is able to either read test vector values from onboard memory or generate them with an embedded ATPG module. We determined that only two vector sources would be required. We created on-board memory to store vectors generated with TetraMAX. We only created an ATPG module for Design A. It consisted of a simple 8-bit up-counter. It provided comprehensive test sets through iterative exhaustion. It is impractical for large designs, but was efficient and effective for smaller designs with fewer than 10 inputs.

3.1.3 Testbed Assembly

The first step was to setup and connect the testbed. Setting up the physical components was fairly straightforward. Jumper leads were used to connect the PMOD pins of the Basys2 board to the I/O headers on the Virtex 5. Design B uses one of the pins to output a reference clock signal; the corresponding input on the Virtex 5 was selected for minimal latency using the Xilinx PlanAhead tool [72]. The rest of the pins were selected arbitrarily, and the pin locations were mapped to the controller design accordingly. A power lead with an inline 10Ω resistor supplied the Basys2 from the Virtex 5. The oscilloscope probes were connected on either side of the resistor for differential measurements. The logic analyzer leads were simply piggybacked onto the Basys2 I/O leads. The resulting system is shown in Figure 6.

Two controllers were constructed, one for testing the combinational design and another for testing the sequential design. The designs are different enough that no part of the controller for one design could be recycled for the other design.
3.2 Execution

First the golden models were analyzed with Synopsys TetraMAX, and full coverage test vector sets were generated. Then, we performed functional verification with the reduced test vector set. The set was generated to provide full fault coverage, which we determined would give us the best chance of activating a Trojan.

The initial functional verification phase was performed with the reduced sets. If they failed to activate a functional Trojan, the test was switched to applying comprehensive vectors. The set was repeated continuously for use in collecting power measurements, as discussed in Section 2.3.4. These measurements were taken during runtime by generating an image file of the waveforms produced by the oscilloscope. Measurements were taken for 512 cycles of testing, and the values were averaged.

If power analysis did not produce a result, we analyzed the functional profiling data. This was only required in one case, as discussed in Section 3.3.6. If all else failed, we used the soft method described in Section 2.3.5. The experiment was designed so that it only had to be run once to collect all the results; we didn’t have to implement a new test if one phase failed.
3.3 Results

As discussed earlier, we were given a set of designs and were required to determine which designs had Trojans. The Trojan either changed the functionality of the design or caused an increase in power in order to reduce reliability. The effects of each Trojan are listed in Table 1. In this section we present and discuss the results of the experiment. Supporting images are contained in the appendices, and their references are prefixed with a letter indicating the appendix. For example, Figure A-11 refers to Figure 11 in Appendix A. The outcome of the Embedded Systems Challenge served to independently verify the correctness of the results presented here. The red team confirmed that we had correctly classified all of the provided designs.

<table>
<thead>
<tr>
<th>Trojan</th>
<th>Function</th>
<th>Reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 1</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>A 2</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>A 3</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>A 4</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>A 5</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>A 6</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>B 1</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>B 2</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

3.3.1 Design A Overview

Design A is a large c6288 combinational circuit from the ISCAS’85 benchmark set with around 2500 gates. The number of I/O ports for the design exceeded the Basys2’s general-purpose input/output (GPIO) capabilities, so a multiplexer system was added to expand the inputs and compress the outputs\(^8\). The synthesized design occupies 101 LUTs on the FPGA [73]. A summary of the results for Design A are provided in Table 2.

\(^8\)The Trojan was designed for the multiplexed system, so the added I/O logic has negligible effect on the ability to detect of the Trojan.
Table 2: Design A Results

<table>
<thead>
<tr>
<th>Trojan</th>
<th>Metric</th>
<th>Trigger</th>
<th>Measured</th>
<th>Expected</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 1</td>
<td>Functional</td>
<td>Input 0x05</td>
<td>Output 0x2D</td>
<td>Output 0x3D</td>
</tr>
<tr>
<td>A 2</td>
<td>Functional</td>
<td>Input 0x96</td>
<td>Output 0x49</td>
<td>Output 0x41</td>
</tr>
<tr>
<td>A 3</td>
<td>Functional</td>
<td>Sequence</td>
<td>output[0] = 1</td>
<td>output[0] = f(input)[0]</td>
</tr>
<tr>
<td>A 5</td>
<td>Power Consumption</td>
<td>Unknown</td>
<td>50mA</td>
<td>45mA</td>
</tr>
<tr>
<td>A 6</td>
<td>Power Consumption</td>
<td>Unknown</td>
<td>48mA</td>
<td>45mA</td>
</tr>
</tbody>
</table>

3.3.2 Design A Trojan 1 and Trojan 2

Trojans 1 and 2 are input-triggered Trojans that alter the functionality of the device, producing incorrect output values for one or more input values. The Trojans are combinational designs and do not have any internal state information. Neither Trojan required any input sequence for activation, enabling detection on the first round of functional testing.

Both Trojans were detected using either the exhaustive set or the TetraMAX-generated set. Applying an input value of 0x05 to the circuit should produce the output 0x3D. Designs compromised with Trojan 1 produced an output value of 0x2D (Figure B-1). Similarly, an input value of 0x96 should result in an output value of 0x41. Designs compromised with Trojan 2 produced the incorrect output 0x49 (Figure B-2). The Trojans may also modify the function for additional input values, but we did not attempt to verify them. We were already able to classify the design based on the single fault, so detecting additional triggers will provide no new relevant information.

3.3.3 Design A Trojan 3 and Trojan 4

Trojans 3 and 4 also change the functionality of the design, but they require a specific sequence of input values before activating. Both designs were detected in the third round of exhaustive functional testing (Figures B-3 & B-5). Neither Trojan was activated when using the reduced vector set.

When activated, the Trojans force one output bit to a logical 1. Trojan 3 affects out[0] and Trojan 4 affects out[4]. Both designs failed functional verification just after the Trojans were activated. Trojan 3 caused a failure when applying 0xCF as the input gave
a result of 0x51 instead of the expected 0x50 (Figure B-4). Similarly, Trojan 4 caused an incorrect 0x61 output for the input 0x3D; the correct output is 0x41 (Figure B-6).

The Trojan activation sequences are discussed in [73]. The author indicates that the activation sequence for Trojan 3 requires four applications of the value 0xCD. This contradicts the behavior of the Trojan, which indicates that it is triggered by only three applications of the value. The activation sequence for Trojan 4 consists of 4 values: 0xB8 \rightarrow 0x63 \rightarrow 0xE4 \rightarrow 0x3D. This matches the behavior of the design; the first value is applied in the first cycle, the second and third values are applied in the next cycle, and the design finally fails when the last value is applied in the third cycle.

3.3.4 Design A Trojan 5 and Trojan 6

Trojans 5 and 6 do not affect the functionality of the design. Their purpose is to increase the power consumption of the design in order to raise the temperature of the IC [73]. The increased temperature accelerates the aging of the transistors, reducing the lifespan of the IC [9]. Since the effects are not visible on the output, neither Trojan was detected by functional verification. Statistical analysis of the power measurements revealed trends which led to the detection of both Trojans.

Trojan 5 caused a significant increase in average power consumption. One set of designs drew an average of 45 mA (Figures A-2 & A-4), while the other set drew 50 mA on average, an increase of about 11% (Figures A-1 & A-3). The results clearly indicated the presence of a Trojan in the design. The effect of Trojan 6 was slightly more subtle. The average power draw increased from 45 mA (Figures A-7 & A-8) to about 48 mA (Figures A-5 & A-6), an increase of about 6%. Again, the average power draw indicates the presence of a Trojan within the design.

3.3.5 Design B Overview

Design B is a large sequential circuit, s9234 from the ISCAS’89 benchmark set. It contains 200 flip-flops and 5000 combinational gates and uses 542 LUTs on the FPGA. Similar to Design A, the inputs and outputs are multiplexed for compression/expansion. It operates at 5 MHz and uses an internal clock source. A scan-chain consisting of 145
flip-flops is inserted which can be used to read or set internal state data during testing [73].

A summary of the results for Design B are provided in Table 3.

Due to hardware limitations of the Basys2 board, we were unable to collect definitive evidence of the presence of a Trojan in either of the designs. We were not able to synchronize the two designs, hindering our ability to perform differential verification; when the golden model and UUT get out of sync, all future data immediately becomes invalid. Section 3.5 provides a discussion of the limitations and their effect on test reliability, along with a brief discussion of possible solutions.

<table>
<thead>
<tr>
<th>Trojan</th>
<th>Effect</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>B 1</td>
<td>Change functionality</td>
<td>High-resolution functional profiling</td>
</tr>
<tr>
<td>B 2</td>
<td>Increase power consumption(^1)</td>
<td>File hash analysis</td>
</tr>
</tbody>
</table>

\(^1\) This effect was not directly observed. See section 3.3.7.

3.3.6 Design B Trojan 1

Trojan 1 changes the functionality of the design. The design description in [73] indicates that the Trojan uses an inverter to modify an output pin value. The automated detection method was not reliable enough for use due to the previously mentioned synchronization issues. We modified the control unit so that it is only used to apply the test inputs. The logic analyzer was used to collect waveforms for visual analysis and verification.

We created a simulation to apply the test vectors to the golden model, and visually compared the behavior for inconsistencies. Fortunately, we were able to distinguish a glitch in one of the output signals where the signal was briefly inverted, indicating incorrect functionality. Normal operation is illustrated in Figures C-1 and C-3, and the glitch is visible on channel 4 in Figures C-2 and C-4. The behavior was confirmed in followup tests and the corresponding designs were classified as Trojans.

3.3.7 Design B Trojan 2

Trojan 2 increases power consumption, but does not affect the functionality of the design. As a result, the Trojan proved to be undetectable by our physical system. The
previously mentioned hardware limitations restricted our ability to control the UUT during the test, which in turn prevented us from obtaining consistent power measurements. Access to a strong reference clock would have enabled us to actually synchronize the UUT, golden model and control unit for proper operation.

Table 4: Design B Results

<table>
<thead>
<tr>
<th>File</th>
<th>MD5 Hash</th>
<th>Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-1-1.bit</td>
<td>93747c0d94b151c9266a0ad7d570</td>
<td>Verified Clean</td>
</tr>
<tr>
<td>B-1-2.bit</td>
<td>45c567f5e25a3fe78a0faef67a99841</td>
<td>Detected Trojan</td>
</tr>
<tr>
<td>B-1-3.bit</td>
<td>45c567f5e25a3fe78a0faef67a99841</td>
<td>Detected Trojan</td>
</tr>
<tr>
<td>B-1-4.bit</td>
<td>93747c0d94b151c9266a0ad7d570</td>
<td>Verified Clean</td>
</tr>
<tr>
<td>B-2-1.bit</td>
<td>883068c672114407d7c2fcd5b62e862f</td>
<td>Compromised</td>
</tr>
<tr>
<td>B-2-2.bit</td>
<td>93747c0d94b151c9266a0ad7d570</td>
<td>Clean</td>
</tr>
<tr>
<td>B-2-3.bit</td>
<td>93747c0d94b151c9266a0ad7d570</td>
<td>Clean</td>
</tr>
<tr>
<td>B-2-4.bit</td>
<td>883068c672114407d7c2fcd5b62e862f</td>
<td>Compromised</td>
</tr>
</tbody>
</table>

We came up with an alternative classification method to compensate. We used the method described in Section 2.3.5 to analyze the bitfiles we were provided. The results, shown in Table 4, indicate that two of the designs match known clean designs. The other two are unique, implying that they are compromised.

When Xilinx Bitgen works its magic, it embeds a timestamp in the resulting bitfile. This timestamp indicates the compilation time, and is represented in ASCII [74]. We had to alter this timestamp in order to do the hash comparison. The output of the MD5 hash is determined by the entire binary representation of the file [75]. A single bit difference between two files will result in different hashes. This caused an issue when comparing the bitfiles. We were able to change the embedded timestamps so that they were the same, enabling us to perform a direct comparison. Changing the timestamps had no effect on the functionality of the design.

3.4 Technique Evaluation

Overall, the proposed technique has proven to be decently robust and flexible. It was able to detect both reliability and functionality modifying Trojan designs.
3.4.1 Advantages

The system described here offers several advantages over other methods. The system is built around an FPGA platform that can be configured to test a range of different designs. Its reconfigurable nature makes it highly flexible; it can scale to test designs of variable size and complexity and can be reconfigured to enable additional tests.

It is possible to use the same hardware setup to test a range of different designs with minimal reconfiguration by simply loading a new control unit and UUT; the I/O pins can be configured to match the existing interconnects.

Built-in redundancy reduces the potential for a false positive or negative by confirming across multiple systems. Additional test systems can be added to increase the range of tests it can perform. Every part of the system is modular, enabling components to be added, removed, or swapped out.

On-the-fly vector generation and differential verification capabilities allow tests to be easily modified without the overhead of generating and verifying new test sets. Due to the dynamic nature of these features, they are really only practical on FPGA systems.

3.4.2 Disadvantages

The applicability of the proposed method to a certain design is limited by the size of the FPGA device it is to be instantiated on. It must be large enough to contain the control unit, test vector sources, and golden model. Very large complex designs may require more resources than are available in the current implementation.

The UUT is run in parallel with the golden model, making precise timing an absolute necessity. The designs must be synchronized to ensure the results are valid. If they fall out of sync, the control unit will attempt to compare unmatched data, which will cause the test to fail. This limitation caused a major issue when testing the sequential design, which is detailed in Section 3.5. This is only a problem if there is no clean reference clock available, which is rarely the case in designs built for testing.
3.5 Design B Clock Complications

When testing sequential designs, the test system is entirely dependent on the availability of a stable and clean reference clock signal from the device. The controller relies on the reference clock to load scan-chain values, apply test input values, and verify output values.

This limitation severely reduced the testability of the given sequential design. The design relied on an internal clock net and a soft clock divider. A flaw in the FPGA design initially prevented the acquisition of a stable clock signal. Some component in the sequential design seemed to conflict with a test program on the Basys2 board.

![Figure 7: 50MHz Clock Superimposed on 5MHz Clock](image1)

![Figure 8: Clock Signal (Yellow) Behavior After Reset (Blue)](image2)

The Basys2 has the ability to automatically load a design file from an onboard ROM, provided the board is configured properly. When shipped, it comes with a basic test program.
that automatically loads when the board is initialized. This program is overwritten when loading a new design. Some part of the test design somehow conflicted with the sequential circuit design, causing a failure in either the clock divider or I/O routing. It resulted in a 50MHz\textsuperscript{9} clock superimposed on a 5MHz\textsuperscript{10} clock, as seen in Figure 7.

The sequential design provided an ability for asynchronous reset of the entire design. This reset signal also performed a reset on the clock divider. This prevented the acquisition of a reference clock for some time after design initialization. Furthermore, after the clock divider resumes operation, the reference clock is completely unstable (Figure 8). The behavior prevented the capture of reliable information during the design initialization phase.

![Figure 9: Clock Signal Subject to Possible Feedback Loop](image)

A final setback resulted from a physical limitation of the FPGA hardware. An FPGA device typically has dedicated nets and IO buffers specifically designed for use with clock signals. They minimize skew and delay, providing a clean clock signal. Neither the Virtex board nor Basys board have any user-accessible clock buffers, forcing the designer to use a general purpose IO buffer to input or output the clock signal. In this case, the reference clock had to pass through two GPIO buffers, one on the Virtex and one on the Basys. The IO buffers are designed with signal stability in mind, rather than minimal latency. General purpose IO buffers are able to function as either inputs or outputs. As a result, the board was highly sensitive to feedback loops and reverse biasing in addition to

\textsuperscript{9}The native frequency of the onboard clock generator.
\textsuperscript{10}The desired frequency of the output from the clock divider.
signal degradation. The effect provided unusable signals, such as the one in Figure 9. The specific cause of some clock failures could not be determined, due to a lack of access to the original design files.

3.6 Chapter Summary

The system we designed was successful in detecting the provided Trojans. It proved most effective when testing combinational designs. The sequential designs were more difficult, but not impossible. Hardware complications prevented us from effectively analyzing the sequential design with the physical components, but the soft analysis method was more than sufficient to make up for it. Overall, the experiment was a success and shows that the system is effective at detecting simple or Trojans in larger FPGA designs.
4 Conclusions and Future Work

Section 4.1 concludes this thesis, and Section 4.2 highlights our plans for future development on this method.

4.1 Conclusions

The system successfully accomplished its intended goal: it correctly classified each provided design as Trojan or clean. These results were verified at ESC by a panel of judges from industry and academia. They judged our design on its abilities and practical uses, and determined it to be both flexible and robust.

The system has proven effective in accurately and rapidly detecting function-altering or reliability-degrading Trojan designs. Once the control unit design had been finalized and implemented, we were able to test and correctly classify all 24 Design A variations within a few minutes. Due to the issues discussed in Section 3.5, we were unable to collect clear, consistent results from one of the Design B Trojans.

The supplemental soft analysis method makes up for the physical shortcomings. We used it to verify nearly every single design we classified\textsuperscript{11}. It may not have much practical application in many cases, but we clearly demonstrated that it can be effective for some situations.

Our first-place win at ESC proved the validity and robustness of the system. The method is solid, reliable, flexible, and effective. Overall, the project and competition were a complete success, and the resulting system has much potential for future development.

4.2 Future Work

The system has a lot of potential for future development. The modular nature of the system allows us to add or remove components to easily modify its abilities.

\textsuperscript{11}Some of the provided files were re-synthesized which prevents accurate soft analysis.
We intend to automate the entire testing process. Test vectors are applied and functional verification is performed, like the current implementation, and the test vectors will be generated at runtime. The control unit will monitor the conditions of the test and analyze the performance of the design. If the current ATPG algorithm is producing poor results, the control unit will compensate with a different algorithm. The challenge will be in creating a heuristic algorithm capable of adapting in such a way.

Automation of the power analysis and functional profiling phases will enhance the efficiency of the test. The data can be analyzed in real-time, similar to functional analysis. Proper implementation of this method will require an overhaul of the control unit. Currently, it does not provide any way to perform real-time power comparison. The golden model is instantiated on the same piece of silicon as the control unit, so it cannot be isolated for power analysis.

One possibility is to create a custom printed circuit board (PCB) with three FPGA devices. The control unit will be configured on a large device, such as a Xilinx Virtex-series device. The UUT and golden model will be configured onto smaller FPGA, such as Spartan devices. The interconnects use matched-length traces to provide consistent delay. Headers will be inserted and connected to I/O, power and clock nets for easy access. This system will provide a foundation for real-time comparative verification of power and delay measurements.
References


49


Appendix A  Oscilloscope Measurements

Figure A-1: Scope Measurements for Design A, Trojan 5, 1.bit

Figure A-2: Scope Measurements for Design A, Trojan 5, 2.bit
Appendix A (Continued)

Figure A-3: Scope Measurements for Design A, Trojan 5, 3.bit

Figure A-4: Scope Measurements for Design A, Trojan 5, 4.bit
Figure A-5: Scope Measurements for Design A, Trojan 6, 1.bit

Figure A-6: Scope Measurements for Design A, Trojan 6, 2.bit
Appendix A (Continued)

Figure A-7: Scope Measurements for Design A, Trojan 6, 3.bit

Figure A-8: Scope Measurements for Design A, Trojan 6, 4.bit
Appendix B  Waveforms

### Figure B-1: Output Values for Trojan A1

| 1000 | 1011 | 10101000 | 10101001 | 10101010 | 10101011 | 10101100 | 10101101 | 10101110 | 10101111 | 10110000 | 10110001 | 10110010 | 10110011 | 10110100 | 10110101 | 10110110 | 10110111 | 10111000 | 10111001 |
|------|------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|

### Figure B-2: Output Values for Trojan A2

### Figure B-3: Test Overview for Trojan A3
Appendix B (Continued)

Figure B-4: Output Values for Trojan A3

Figure B-5: Test Overview for Trojan A4

Figure B-6: Output Values for Trojan A4
Appendix B (Continued)

Figure B-7: Power Measurements for Trojan A5

Figure B-8: Power Measurements for Trojan A6
Appendix C  High-Resolution Output Waveforms

Figure C-1: High-Resolution Output Capture for Design B, Trojan 1, 1.bit

Figure C-2: High-Resolution Output Capture for Design B, Trojan 1, 2.bit
Appendix C (Continued)

Figure C-3: High-Resolution Output Capture for Design B, Trojan 1, 3.bit

Figure C-4: High-Resolution Output Capture for Design B, Trojan 1, 4.bit
ASCII
American Standard Code for Information Interchange; a standardized character-encoding format.

ASIC
Application-Specific Integrated Circuit; an integrated circuit custom designed for a specific use, rather than general-purpose use.

ATPG
Automatic Test Pattern Generation; method used for generating a sequence of input and output values for use in testing a digital circuit.

Basys2
An FPGA prototyping board developed by Digilint.

bitstream
Digital representation of a circuit, used to configure FPGA devices.

Chipscope
A diagnostic tool used to monitor and debug designs on Xilinx FPGA devices.

CMOS
Complementary Metal-Oxide Semiconductor; the most common commercial technology for constructing integrated circuits.

combinational
A logic circuit in which the output is a pure function of the current input only, regardless of previous inputs.

COTS
Commercial, Off-The-Shelf; a ready-made component or item that is commercially available by the general public.

CPLD
Complex Programmable Logic Device; a field-programmable logic device that stores configuration information in non-volatile memory. Unlike a PLA, the configuration can be overwritten, allowing the device to be reconfigured multiple times.
Appendix D (Continued)

CSAW
Cyber Security Awareness Week, an annual event hosted by NYU-Poly focusing on modern cybersecurity issues.

CSV
Comma-Separated Value; a digital file format used for store organized data in a simple tabular format.

DDFV
Dynamic Differential Functional Verification; a proposed method of functional verification that uses direct comparison between a UUT and its golden model for real-time dynamic testing.

Digilint, Inc.
A US-based company which builds, sells, and distributes complete FPGA evaluation platforms.

EoL
End of Lifetime; refers to a product which no longer receives vendor support and is no longer being actively produced for sale.

ESC
Embedded Systems Challenge, an event hosted as part of CSAW focusing on the hardware aspect of cybersecurity.

fabless
Referring to an IC design company that does not operate its own IC fabrication facilities.

fabrication
The process of manufacturing, or fabricating, a physical IC.

field-programmable
A device or system that can be programmed ‘in-the-field’ without requiring disassembly or physical replacement.

foundry
A facility that provides commercial IC fabrication services to companies or individuals.
Appendix D (Continued)

FPGA

Field Programmable Gate Array; a field-programmable logic device that stores configuration information in volatile memory. Unlike a CPLD or PAL, the configuration information is lost when the device loses power.

FSM

Finite State Machine; a mathematical computation model describing the function and execution of a sequential system.

functional verification

The process of testing a device to verify that it functions as expected.

GLC

Gate-Level Characterization; a model used to extrapolate the expected physical characteristics of a fabricated design from its logical description.

glitch

In digital logic, an undesired signal transition with a very short duration. The event is typically too brief to interrupt normal operation.

golden model

A representation or model of a system which is known to be both functional and correct.

GPIO

General-Purpose Input/Output; an input or output of a system which is not used for a specific purpose. On an FPGA board, only GPIO ports are accessible for use by the user, other I/O ports are dedicated to other systems.

GS/s

Giga-Samples per Second; used to indicate how many times a signal is sampled, in billions.

GUI

Graphical User Interface; an interface which allows a user to interact with software.

HDL

Hardware Description Language; a computer language used to describe the function, behavior or structure of digital systems.
Appendix D (Continued)

HTH

Hardware Trojan Horse; a malicious modification to a hardware design. The name is a reference to the Trojan Horse story from Greek mythology.

I/O

Input/Output; a physical connection through which an embedded system can communicate with an external system.

IC

Integrated Circuit; a set of electronic circuits that are manufactured on a single piece of material.

IDE

Integrated Design Environment; a software environment that provides a comprehensive set of tools necessary for a development process.

IEEE

Institute of Electricians and Electrical Engineers; a professional organization dedicated to advancing technological innovation and excellence. IEEE maintains many engineering standards related to electronics and electronic devices.

IP

Intellectual Property; a design or information that has its usage and application strictly controlled by the owner for financial, security, or other purposes.

ISCAS

The IEEE International Symposium on Circuits and Systems; originating source of base design files used to embed Trojans into.

ISCAS’85

Set of HDL designs used as standard benchmarks for relevant works presented at the ISCAS 1985 conference.

ISCAS’89

Set of HDL designs used as standard benchmarks for relevant works presented at the ISCAS 1989 conference.

ISim

The integrated simulation environment provided as an element of the Xilinx ISE, used to simulate HDL code.
Appendix D (Continued)

IVI

Interchangeable Virtual Instrument; An industry-standard PC interface protocol for scientific instruments maintained by the IVI Foundation.

JTAG

Joint Test Action Group; organization that maintains various test protocol standards, including the JTAG cable programming protocol.

JTAG cable

A cable used to connect an FPGA device to a PC for configuration and monitoring; uses a protocol standardized by JTAG to provide bi-directional communication.

LA4850

A legacy logic analysis platform formerly produced by Link Instruments; Current status is EOL.

LED

Light Emitting Diode; a solid-state device that emits light when voltage is applied.

LUT

Look-up table; a memory structure used in FPGA architecture to emulate logical blocks.

Matlab

MATrix LABoratory, A numerical computing environment developed by MathWorks.

MD5

Message-Digest 5; a cryptographic hash function that produces 128-bit hashes.

Minitab

Statistical analysis tool developed by Minitab, Inc.

Mixed Signal Oscilloscope

An oscilloscope that has both analog and digital signal capture capabilities.

ML509

Virtex 5 evaluation platform model which contains a Virtex 5 LX110T FPGA device. Distributed through the Xilinx University Program. See XUPv5.

MSO2014

Digital mixed-signal oscilloscope produced by Tektronix with analog and digital signal capture and analysis capabilities.
Appendix D (Continued)

NYU-Poly
The Polytechnic Institute of New York University; A New York university located in Brooklyn, New York City, NY.

on-the-fly
An operation or action performed as required at runtime, as opposed to preemptively during preparation phases.

PC
Personal Computer; used to refer to a general-use computer system.

PCB
Printed Circuit Board; a manufactured component used to construct electronic devices, it contains routing and connections for IC components.

PLA
Programmable Logic Array; a field-programmable logic device that stores configuration information in an embedded PROM.

PlanAhead
FPGA design tool for optimizing physical constraints during the design process. Part of the Xilinx ISE.

PMOD
Physical component interface specification maintained by Digilent for use with its products; refers to the four 6-pin connectors on the Basys2 board.

power analysis
The process of measuring the power consumption of a device in order to determine details about the construction of the device.

process variation
Naturally occurring variation of the attributes of transistors during IC fabrication.

PROM
Programmable Read-Only Memory; a field-programmable ROM device used to permanently store data. Unlike a ROM, the programming phase is separate from the manufacturing phase.

ROM
Read-Only Memory; A memory device on which data is permanently written during fabrication.
Appendix D (Continued)

SCA
Side-Channel Analysis; the method of using physical implementation characteristics of a design to determine functional characteristics.

scan-chain
A structure included on some sequential circuits, often used to enhance or ease the process of testing.

sequential
A logic circuit in which the output is a function of both the current input and previous inputs.

side-channel analysis
Analysis of a system based on information gained from its physical characteristics rather than functional behavior.

SOC
System-on-Chip; a self-contained system of multiple components and interconnects constructed on a single IC.

Synopsys, Inc.
A US-based company which provides a range of software tools used for FPGA and IC design and development.

test vector
A set of input and corresponding output values used to verify the function of a system.

Testbed
An isolated system or set of components used as a prototyping or experimentation platform in research.

TetraMAX
Automated HDL testing software from Synopsys that is able to automatically generate test vector sets for a provided design.

Trojan
See Hardware Trojan Horse.

USB
Universal Serial Bus; a standard PC peripheral interface protocol supported by most PC-capable devices.
Appendix D (Continued)

USF
The University of South Florida; a Florida university with a main campus in Tampa and satellite campuses throughout West-Central Florida.

UUT
Unit Under Test; the physical device being verified.

$V_{DD}$
Voltage Drain-to-Drain; refers to the main supply power for a device.

Verilog
A weakly-typed HDL programming language, used to create FPGA designs for Xilinx devices.

VHDL
A strongly-typed HDL programming language, used to create FPGA designs for Xilinx devices.

Virtex 5
An advanced FPGA platform produced by Xilinx.

waveform
A graphical representation of numerical data; specifically a value-vs-time domain graph of a digital logic signal.

WaveView
Software solution from Synopsys used to view and analyze waveforms.

Xilinx ISE
An IDE for designing and analyzing FPGA systems, developed by Xilinx for use with their FPGA products.

Xilinx, Inc.
A US-based company which develops and produces reconfigurable logic devices and development tools.

XUPv5
An advanced Virtex 5 FPGA evaluation platform made available to universities by Xilinx through its Xilinx University Program.