AFRL-RI-RS-TR-2022-045

PCB-LEVEL TROJAN INSERTION SYSTEM

UNIVERSITY OF FLORIDA

MARCH 2022

FINAL TECHNICAL REPORT

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

FOR THE CHIEF ENGINEER:

/ S /
SERGEY PANASYUK
Work Unit Manager

/ S /
JAMES S. PERRETTA
Deputy Chief
Information Warfare Division
Information Directorate

REPORT DOCUMENTATION PAGE

1. REPORT DATE MARCH 2022	2. REPORT TYPE FINAL TECHNICAL REPORT	3. DATES COVERED START DATE DECEMBER 2020	END DATE DECEMBER 2021

4. TITLE AND SUBTITLE
PCB-LEVEL TROJAN INSERTION SYSTEM

5a. CONTRACT NUMBER N/A	5b. GRANT NUMBER FA8750-21-1-1001	5c. PROGRAM ELEMENT NUMBER 62716E
5d. PROJECT NUMBER	5e. TASK NUMBER	5f. WORK UNIT NUMBER R35X

6. AUTHOR(S)
Swarup Bhunia

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Florida Benton Hall 325 968 Center Drive Gainesville FL 32611	8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RIGB 525 Brooks Road Rome NY 13441-4505	10. SPONSOR/MONITOR'S ACRONYM(S) RI	11. SPONSOR/MONITOR'S REPORT NUMBER(S) AFRL-RI-RS-TR-2022-045

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Printed Circuit Broad (PCB) level tampering attacks in an untrusted supply chain pose a major threat to DoD electronic systems. These tampering attacks, also known as PCB-level Trojans, are extremely challenging to detect due to the growing complexity of commercial COTS systems and lack of golden designs. In this grant project, we have made significant progress on developing preliminary versions of a board-level Trojan benchmarking platform and an innovative board-level Trojan detection tool.

15. SUBJECT TERMS
Trojan, PCB, Supply Chain

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT U	b. ABSTRACT U	C. THIS PAGE U	SAR	57

19a. NAME OF RESPONSIBLE PERSON SERGEY PANASYUK	19b. PHONE NUMBER *(Include area code)* N/A

PREVIOUS EDITION IS OBSOLETE. STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

TABLE OF CONTENTS

List of Figures iii

List of Tables v

1.0 SUMMARY 1

2.0 INTRODUCTION 2

3.0 Related Works 4

4.0 An Automated Framework for PCB-level Trojan Benchmarking 5

4.1 LIFE CYCLE OF PCB 5

4.2 DEFINITIONS 6

4.2.1 PCB-LEVEL TROJAN 6

4.2.2 PCB TROJAN BENCHMARKS 6

4.3 PCB TROJAN TAXONOMY AND TROJAN DESIGNS 6

4.3.1 PCB TROJAN TAXONOMY 6

4.3.2 TROJAN DESIGNS 8

4.4 Toolflow for Synthetic Trojan Benchmark 11

4.4.1 SELECTION OF TROJAN AND INSERTION LOCATION 13

4.4.2 DYNAMIC VARIANT CREATION 15

4.4.3 POST-INSERTION VERIFICATION AND ANALYSIS 16

4.4.4 FABRICATION 18

4.5 RESULTS 19

4.5.1 EXPERIMENTAL RESULTS FROM FABRICATED PCB 19

4.5.2 PCB TROJAN BENCHMARK GENERATION 21

4.5.3 BENCHMARK DIVERSITY AND TROJAN STEALTHINESS 21

4.5.4 COMPLEXITY ANALYSIS OF THE TOOL 23

5.0 An Automated Framework for PCB-level Trojan Benchmarking 25

5.1 PCB-LEVEL TROJAN INSERTION EMULATOR 25

5.2 PCB-level Trojan Models on the Custom HaHa Board 26

5.3 PCB-LEVEL TROJAN MODELS ON COMMERCIAL DESKTOP MOTHERBOARDS 27

5.4 EXPERIMENTAL SETUP AND DEMONSTRATION 28

5.4.1 PCB TROJAN IMPLEMENTATIONS ON CUSTOM HAHA BOARDS 28

5.4.2 PCB TROJAN IMPLEMENTATIONS ON COMMERCIAL BOARDS 28

5.5 SETUP FOR EXPERIMENTAL MEASUREMENTS 32

5.6 TROJAN ACTIVATION AND VALIDATION RESULTS 33
5.6.1 HARDWARE TROJAN INSERTION RESULTS ON THE HAHA BOARD...... 33
5.6.2 HARDWARE TROJAN INSERTION RESULTS ON THE HAHA BOARD...... 35
5.6.3 HARDWARE TROJAN INSERTION RESULTS ON COMMERCIAL BOARDS 35
5.7 LIMITATIONS 37
5.8 FLEXIBILITY AND SCALABILITY.. 37
5.9 INTEGRATION WITH EXISTING PCB ANALYSIS TOOLS.................................. 38
6.0 A Machine Learning based Golden-Free PCB Assurance Framework 39
6.1 TROJAN DETECTION METHODOLOGY 39
6.2 TRAINING DATASET GENERATION.. 39
6.3 DATASET PRE-PROCESSING.. 40
6.4 FEATURE EXTRACTION 40
6.5 TROJAN DETECTION IN SUSPECTED PCB 41
6.6 EXPERIMENTS AND RESULTS 42
6.6.1 BOARD LEVEL ANALYSIS (BLA) 42
6.6.2 ELEMENT LEVEL ANALYSIS (ELA) 42
6.6.3 TRAINING DATA REQUIREMENT ANALYSIS.. 42
7.0 CONCLUSIONS.. 45
8.0 References.. 47
LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS... 49

LIST OF FIGURES

Figure 1. Taxonomy of PCB-level hardware Trojans. The dotted-box show the Trojan classes primarily applicable to PCBs. 6

Figure 2. High-level illustration of addition model Trojan that is designed by introducing additional com-ponent in the PCB. We present a number of methods to construct hard-to-activate trigger logic and payload logic to leak a victim signal or corrupt its function 7

Figure 3. Three different implementation of 2 input NAND: (a) using resistor-transistor logic, (b) diode-transistor logic, and (c) transistor-transistor logic. 9

Figure 4. Example of alteration model: signal from component 3 with higher drive strength can overdrive the signal going from component 1 to component 2. Drive strength can be increased through the alteration of trace impedance or structure. 9

Figure 5. Algorithm for Trojan Inertion. 11

Figure 6. (a) Trigger circuit similar to A2 Trojan [14] where capacitor and op-amps are used to design activation mechanism that requires the rare condition to occur multiple times. Payload with single transistor: (b) impacts the victim when the transistor is off, (c) impacts the victim when the transistor is on. 12

Figure 7. Algorithm for Trojan Variant Generation of Discrete Templates 15

Figure 8. Layout of the Trojan inserted PCB using KiCAD 17

Figure 9. Snapshot of error message in KiCAD while opening a Trojan inserted netlist with invalid connec-tions. 17

Figure 10. (a) Routed PCB layout obtained from the netlist after Trojan insertion using the tool. (b) 3-D view generated from the same netlist in KiCAD with the Trojan components marked in red. 18

Figure 11. Fabricated PCB Trojan platform designed from our benchmarks 20

Figure 12. Increase in current consumption due to Trojan activation (obtained from fabricated PCB) 21

Figure 13. Figure shows the difference metric for different various pairs of PCB designs. Dark shade indicates lower difference and lighter shades indicate higher difference among two designs. The difference between the Trojan free and Trojan inserted version of the same design (e.g., ciaa_acc vs ciaa_acctr) have very minute difference. Conversely, two different benchmarks have very high difference value (e.g., ciaa_acc vs CIAA_K60). 23

Figure 14. Graph of tool runtime for inserting 50 Trojan instances in host benchmarks of various sizes. The runtime is proportional to the number of nets in the host design. For inserting different Trojans in the same host design, there are minute variations in runtime. 24

Figure 15. A high-level view of the system architecture used for Trojan insertion experiments. 25

Figure 16. Triggers used for emulating PCB-level Trojan insertion using the custom HaHa board: (a) an AND gate, and (b) a NOR gate. Triggers used for emulating PCB-level Trojan insertion using commercial computer motherboards: (c) an AND gate plus a circuit to hold. 26

Figure 17. Trojan payloads used for emulation with the custom HaHa board: (a) an XOR gate, (b) a multiplexer that outputs the victim signal when enabled, and (c) a multiplexer with a linear feedback shift register (LSFR) circuit at the inputs that outputs the victim. 26

Figure 18. Block diagram of Trojans implemented on custom Haha boards that (a) invert the state of an on-board LED, and (b) leak the state of the LED. In both cases, the Trojan is triggered when the user turns on all four on-board switches. 28

Figure 19. Block diagram of an "automatic computer freeze" Trojan triggered by a combination of USB device plug-in and audio playback events. 29

Figure 20. Block diagram of an "automatic CPU fan speed change" Trojan triggered by a combination of USB device plug-in and audio playback events. 29

Figure 21. Block diagram of an "automatic audio data leakage" Trojan triggered by the plug-in of USB devices. 31

Figure 22. Block diagram of an "automatic networking data leakage" Trojan triggered by the plug-in of USB devices. 31

Figure 23. (a) The custom hardware security platform (HaHa board) used for building the trigger and payload circuits. (b)-(d) Trojan insertion victims: (b) a motherboard, (c)-(d) a single-board computer (both front and back sides shown). 32

Figure 24. Experimental setup for emulating Trojan insertion using (a) the HaHa board, and (b) the motherboard. 32

Figure 25. Typical time-domain measurement results of functional leakage-type Trojans: (a) Trojan #3, and (b) Trojan #4. 35

Figure 26. Implementation of the BMC module and other complex trigger logic can be done with the FPGA. 38

Figure 27. Overview of the proposed framework for hardware Trojan detection using Machine Learning. 39

Figure 28. The feature extraction process from sample PCB designs: In (a) and (b) the sample PCB design (top) and the corresponding adjacency matrix (bottom) are shown. The part of the adjacency matrix colored in green is flattened to make the feature vector. 40

Figure 29. Overall detection accuracy across all scenarios for different training dataset sizes with structural features. 43

Figure 30. Overall detection accuracy across all scenarios for different training dataset sizes with both structural and functional features. 44

LIST OF TABLES

Table 1. Examples of Various Real-World PCB-level Trojan Attacks. 10
Table 2. Trigger and Payload of the Fabricated Trojans. 19
Table 3. Overhead Analysis for Trojan Inserted PCB Designs 19
Table 4. Summary of Trojan designs implemented on the HaHa board...................................... 34
Table 5. Risk Level Metrics for Trojans on Commercial PCBs 36
Table 6. Sample PCB Designs for Inserting Trojans 43
Table 7. Trojan detection results for board level analysis 44
Table 8. Trojan detection results for element level analysis 44

1.0 SUMMARY

Printed Circuit Broad (PCB) level tampering attacks in an untrusted supply chain pose a major threat to DoD electronic systems. These tampering attacks, also known as PCB-level Trojans, are extremely challenging to detect due to the growing complexity of commercial COTS systems and lack of golden designs. Existing Trojan detection techniques suffer from the following major deficiencies: (1) they focus on Trojan detection at chip level (e.g., µP, FPGA) and do not scale to board level; (2) board level inspection techniques (e.g., optical inspection) do not scale to large volume, and are expensive, time-consuming, and often not reliable in covering diverse Trojan space; and (3) they rely on availability of golden or reference designs, which may not be available. ***In this seedling project, we have made significant progress on developing preliminary versions of a board-level Trojan benchmarking platform and an innovative board-level Trojan detection tool. We have also studied associated metrics and evaluated these tools for effectiveness with respect to diverse modes of tampering attacks. We have observed that our solutions can effectively*** address the major deficiencies of state-of-the-art approaches and provides a scalable, flexible PCB assurance platform. In particular, the seedling project has enabled us to develop and evaluate:

1. ***A preliminary version of CAD tool to automatically generate large population of synthetic Trojans of all forms and sizes (i.e., different Trojan classes with varying triggers/payloads), for various open-source PCB designs***. We use a graph-based analysis for a PCB design to develop a novel dynamic PCB benchmarking algorithm and tool, that uses parameterized Trojan templates built on viable attack scenarios. The tool is "configurable" by the users, through use of Trojan templates. We have evaluated the tool for 10 open-source PCB designs.

2. An integrated PCB-level Trojan insertion system using a custom PCB built with Intel MAX 10 FPGA. We considered a PC motherboard as target PCB under test. We successfully demonstrated the efficacy of the platform in emulating various Trojans with payloads of denial of service or information leakage.

3. ***Finally, a CAD tool that integrates a Machine Learning (ML) engine to infer existence of Trojan in a PCB from its functional and structural parameters.*** It enables automatically provisioning the ML based inferencing system to account for newly discovered Trojan attacks or new vulnerabilities. It is trained on collected data on sample commercial boards with large number of synthetic Trojans inserted on it. We observe very high detection accuracy achieved by this tool.

2.0 INTRODUCTION

Printed circuit boards (PCBs) are an integral part of the electronic devices' life cycle as they provide essential mechanical support and electrical connections to microchips and discrete onboard components. The economic and operational advantages have led the PCB supply chain to incorporate various untrusted entities. As a result, PCBs are becoming increasingly vulnerable to malicious design alterations, also known as Trojan attacks, due to such distributed business model. Such mischievous modifications can cause functional failures or secret information leakage during field operations.

While researchers have been investigating the threat of malicious modifications within the regime of individual microelectronic components, the possibility of PCB-level manipulations has primarily been unexplored. PCB-level Trojans are extremely challenging to detect due to (1) the lack of golden or reference models in most use cases, (2) potentially unbounded attack space, and (3) the growing complexity of commercial PCB designs. Existing PCB inspection techniques (e.g., optical and electrical) do not scale to large volumes and are expensive, time-consuming, and often unreliable in covering diverse Trojan spaces. In the absence of standard benchmarking solutions, prospective countermeasures for PCB trust assurance are likely to utilize homegrown representation of the attacks that undermine their evaluation and do not provide scope for comparison with other techniques. Additionally, simulation-based approaches for PCB Trojan insertion, while effective at creating a large population of possible Trojans, fail to provide functional feasibility analysis with a realistic workload for a trigger circuit. They are also incapable of estimating a Trojan's side-channel footprint due to unavailability of physical models of diverse PCB components. Correspondingly, there is a growing need to explore possible Trojan attacks on PCBs, analyze their functional and physical characteristics (e.g., impact on power or delay), and study the effectiveness of countermeasures against these attacks.

In this report, we present an extensive approach to address the PCB trust and assurance-related concerns encompassing three critical aspects. Firstly, we develop a software-based benchmarking solution to facilitate an unbiased and comparable evaluation of countermeasures applicable to PCB trust assurance. Based on a taxonomy tailored for PCB-level alterations, we design a software tool flow for automatic generation of Trojan benchmarks in facilitating a comprehensive evaluation against a large number of diverse Trojan implementations and application of data mining for trust verification. Using the tool flow, we develop a fully custom ``Trojan Benchmarks Suite'' (i.e., PCB designs with Trojans) containing representative examples of Trojans in the taxonomy inserted in different PCB designs of varying complexity and functionality. We analyze the Trojan designs ' stealthiness through experimental measurements from a custom-made PCB and a netlist's structural analysis and present the tool's runtime for many PCB designs.

Furthermore, we introduce a PCB-level emulation system for Trojan insertion and activation in actual hardware. It is the first and only existing hardware Trojan emulation platform ever presented to the best of our knowledge. We perform hardware-level validation on both custom-made PCB (Hardware Hacking/HaHa board) and multiple commercial PCBs (e.g., desktop motherboard) to test the proposed system's ability in emulating various hardware Trojans that are designed to exploit board-specific hardware characteristics. Our experimental results demonstrate that a wide range of Trojans can be successfully activated, thus enabling the expected payload effects on both types of boards to be studied and quantified. Additionally, we analyze the resulting data to create PCB-level Trojan benchmarks. We utilize a comparative evaluation of the experimental results to propose a risk-level metric that quantifies the detection probability and degree of payload impact of each Trojan on a given commercial PCB.

Finally, we present a PCB-level Trojan detection framework that employs a machine learning (ML) model to learn Trojan signatures in functional and structural space and uses a trained model to discover Trojans in suspected PCB designs with high confidence level. Using extensive evaluations with ten open-source PCB designs and a wide variety of Trojan instances, we exhibit that the trained model can achieve over 98% accuracy and detect Trojans in partially recovered PCB designs.

3.0 RELATED WORKS

The possibility of PCB-level hardware Trojan was first discussed in [1] with a general model of Trojan attacks applicable to PCB. They also demonstrate two specific attack instances assuming different sets of supply chain entities to be untrusted. However, it does not provide a detailed taxonomy for PCB-level Trojans and does not present any benchmark suite.
An existing work on hardware Trojan taxonomy and benchmarking covers malicious modification of digital IC designs [2]. A benchmark suite containing 96 Trojan-inserted designs are available on the Trust-HUB website. These designs are provided in the form of register-transfer-level (RTL) codes, gate-level netlists, and layouts. The Trojans are contained in various IP cores such as AES, RS232, MP8051, etc. The suite contains few representative Trojan benchmarks from various broad classes of Trojans in the taxonomy.

To eliminate the limitations inherent to a static benchmark suite, [3] proposed a tool-based Trojan insertion process to generate a diverse possible implementation for each class of Trojans. The tool takes a sample gate-level design of an IC and some configuration parameters that define the functional and structural behavior of the Trojans to be inserted. The tool identifies potential regions in the sample design for Trojan insertion and integrates a gate-level Trojan circuit. It also verifies if the Trojan can be activated under a specific input pattern to the sample design. Similar to Trust-HUB Trojans, the tool-based Trojan insertion methods do not serve the benchmarking purpose for PCB-level Trojan attacks. {While the gate-level Trojan insertion tool uses a directed hypergraph to represent the netlist, PCB designs require bi-directional hypergraph data structure which involves different functions for traversal and modification of the graph. Moreover, PCB designs have much higher diversity of components than gate-level designs, which needs to be properly accounted for during graph analysis as well as during simulation. The PCB-level Trojans are more diverse in structure as well, requiring dedicated insertion technique for different Trojan types. Therefore, existing algorithm and tool for gate-level Trojan insertion in IC design cannot be directly used for PCB.}

The authors in [4] presented a taxonomy of PCB-level malicious modification and toolflow for automatic Trojan insertion.
Unlike the PCB Trojan taxonomy in [4], our revised taxonomy provides more consistent nomenclature to widely cited IC-level Trojan taxonomy presented in [2]. It also provides a more fine-grain classification of the attacks compared to [4] that allows a more accurate representation of coverage over various countermeasures. Compared to the PCB-level Trojan insertion tool flow presented in [4], ours proposed technique is widely adaptable by the research community as it is based on a freely-available KiCAD tool. The Trojan-inserted KiCAD netlists are suitable for SPICE simulation and can be converted to PCB layout for fabrication. KiCAD is also compatible with external tools such as Altium (PCB design tool), Eeshow (schematic differences analyzer), fped (repeated footprint automation), KiCad StepUp (integration of mechanical models in KiCAD), KiCost (generating part-cost spreadsheets), and many others [5]. Furthermore, our tool flow can perform dynamic alteration of the existing templates during insertion to create functional variants of the Trojans; enabling the creation of a large pool of diverse Trojans from a limited number of designs. Such a diverse Trojan database is suitable for both evaluations of countermeasures and training ML models for detection.

4.0 AN AUTOMATED FRAMEWORK FOR PCB-LEVEL TROJAN BENCHMARKING

4.1 LIFE CYCLE OF PCB

PCBs are commonly made from non-conductive substrate materials where layers of copper interconnects are etched to draw connections between the components on the board's surface. Not all PCBs go through the exact same steps, but generally, the process of forming a PCB board starts with a concept of the product elaborating its functionality and requirements for the intended application (e.g., area footprint, cost, power budget etc.).

The high-level concept of the product is partitioned into several functional units and for each unit, the design team develops a rough sketch of the schematic that can attain the desired functionality. Some partitions of the schematic can be simulated to ensure that the intended functional and parametric behavior can be attained with the current design. Oftentimes, the schematic is finalized through multiple iterations of redesign and simulation. Once the schematic is completed, it is drawn using software that can further assist in the layout generation process. In many cases, the simulation and actual design process is performed using different software. To finalize the schematic, the design team must ensure that the specific components used in the schematic are readily available in the market. This stage is also known as the Bill of Materials (BoM) validation where the specification, quality, and availability of each component used in the schematic is verified.

After BoM is validated, the layout of the design is created. The layout creation stage takes advantage of an EDA tool that helps to create a board outline, importing netlist (obtained from schematic), placement of netlist components, routing the interconnects, design rule check (DRC), silkscreen cleanup, and generation of final design files for fabrication (e.g., Gerbers, netlists, etc.).

The design files are sent to a fabrication facility where the board containing the interconnect is manufactured. The next stage in the supply chain is the assembly process where the components are mounted on the surface of the board. The assembled PCB is tested to ensure that all the components are placed correctly and the system is providing the correct functionality. Various Design-for-Test structures, such as boundary scan and built-in-self-test units are generally available within the PCB design to facilitate the testing process.

For economic and practical reasons the design, fabrication, and assembly process of PCB is generally done by different vendors. If the design house procures the design from a third-party vendor in the form of a schematic, netlist, or layout, it cannot be trusted. Even if the design is in-house and considered trustworthy, reliance on a third-party fabrication or assembly facility introduces the potential of malicious modifications to the original design. In some cases, a PCB is procured from the market and the consumer is completely isolated from the design and fabrication process. In such a scenario, all the supply chain entities, including the design house, foundry, and assembly and testing facility are considered untrusted. Therefore, in the current business model, there are several scenarios where the trustworthiness of a PCB can be undermined.

4.2 DEFINITIONS

4.2.1 PCB-LEVEL TROJAN

PCBs consist of various microelectronic components spread across multiple alternating layers of conducting and insulating materials. Conductive planes across different layers are connected through vias.

A PCB-level Trojan can be designed by introducing additional components that do not contain any malicious sub-circuit internally. For instance, to construct a PCB-level Trojan, an attacker may introduce several additional NPN transistors that strictly follow the behavior of a generic NPN transistor but do not have any relation in serving the intended functionality of the PCB.

Board-level Trojans could also be introduced without integrating additional components by only altering the structural and/or parametric behavior of conducting (e.g., copper traces, vias, etc.) and insulating parts of the PCB layers.

A more complex attack vector utilizing both intra-board and intra-component modification is possible as well. However, in this report, we assume that the individual microelectronic components perform their designated functionality and there is no malicious functionality hidden *inside* the components.

4.2.2 PCB TROJAN BENCHMARKS

A PCB Trojan benchmark is a PCB design with one or more PCB-level Trojan circuits implanted on it. The benchmark could be in the form of specification, netlist, schematic, or layout.

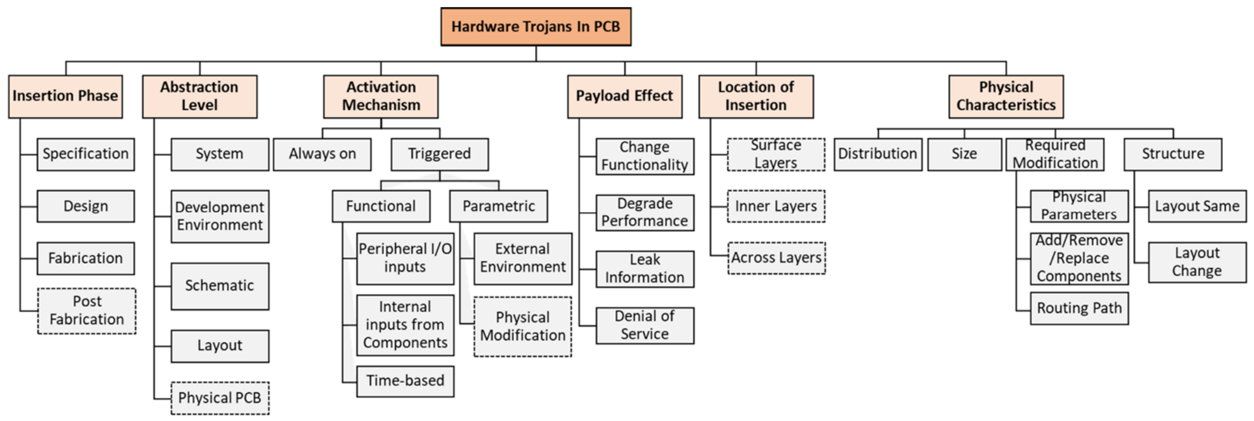


Figure 1. Taxonomy of PCB-level hardware Trojans. The dotted-box show the Trojan classes primarily applicable to PCBs.

4.3 PCB TROJAN TAXONOMY AND TROJAN DESIGNS

4.3.1 PCB TROJAN TAXONOMY

The only existing classification of PCB-level Trojans divides the Trojan space into two broad categories. In the first category, the design house is assumed to be trusted, and in the second one, both design house and foundry are considered to be untrusted [1]. To precisely identify the coverage, strength, and weakness of various countermeasures, a taxonomy of Trojans with fine-grained classification is required. Again, existing taxonomy for IC-level Trojans do not directly apply for PCB as their design abstractions, flexibility for modification, and post-fabrication physical access to the internals are different [2]. Hence, by improving upon the existing taxonomy of IC-level Trojans, we have developed a fine-grained classification for PCB-level Trojans as presented in Figure 1. We can observe several categories of Trojans that are only applicable for PCBs (marked with dotted box). For instance, introducing additional components or alteration of traces after fabrication and assembly of the board is feasible in PCB. Such modifications are usually considered as invasive physical attacks [6]. The goal of PCB physical tampering is to observe an immediate malicious outcome during the attack, such as modchip attacks in gaming consoles. However, Trojan insertion, whether before or after fabrication, is likely to occur with a future goal to compromise the system.

Most in-field invasive attacks on ICs require significant resources for decapsulation, delayering, and modification at the nano-scale level. For PCBs, physical alteration after fabrication is more feasible, which motivates us to consider post-fabrication alteration in the taxonomy. The physical alteration could also be used as an activation mechanism for certain Trojans. For instance, thinning of traces at the surface layer is very feasible and can be used to trigger a failure from heating generated during long hours of operation [1]. The location of insertion in a PCB can be classified with respect to the target layer of alteration. For instance, PCB designs developed by third parties may have malicious circuitry at the surface level of the PCB, as the consumers do not have a reference design. However, in the presence of a reference or "golden" design, the attacker is likely to introduce the alteration within the hidden layers only.

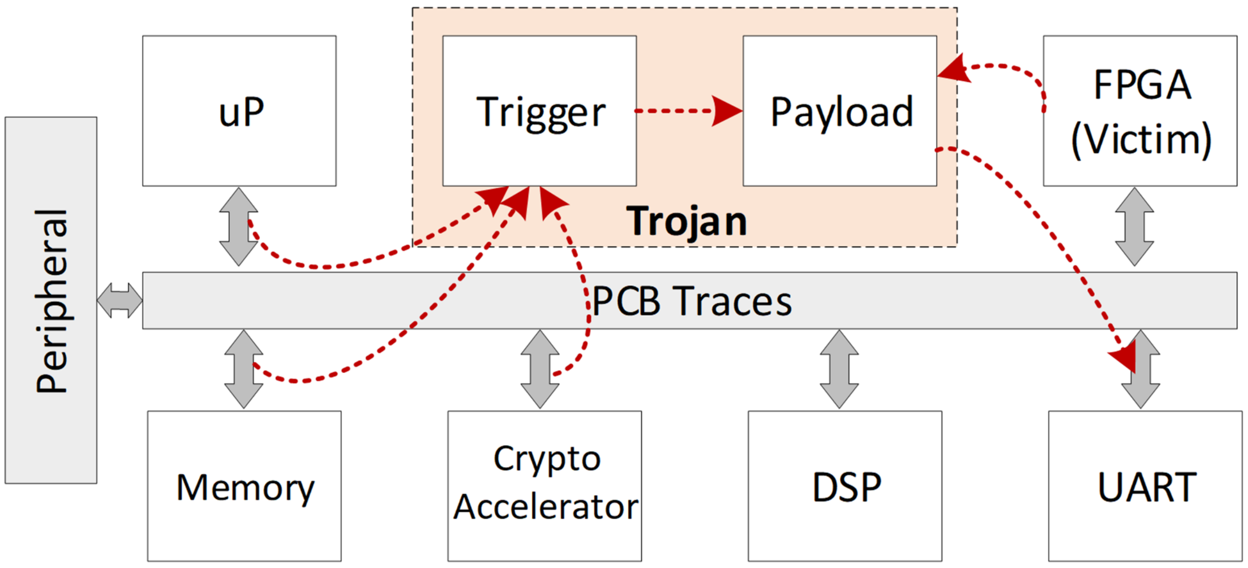


Figure 2. High-level illustration of addition model Trojan that is designed by introducing additional com-ponent in the PCB. We present a number of methods to construct hard-to-activate trigger logic and payload logic to leak a victim signal or corrupt its function

4.3.2 TROJAN DESIGNS

We use two high-level Trojan models to describe the Trojan designs. These two models are defined by the type of change required for insertion.

The designs under these two models represent various Trojan classes within the taxonomy. The first model requires the addition of new components, thus named *Addition Model*. The second model is called *Alteration Model* that alters the existing PCB layout without adding new components.

4.3.2.1 ADDITION MODEL

Figure 2 shows the generic structure of this model that contains a trigger and a payload. The trigger takes one or more inputs from the existing traces of the board to construct a hard to activate trigger condition. This is necessary to ensure that the Trojan does not get accidentally triggered during functional testing of the board. Difficult activation can be ensured by integrating the large number of traces that rarely reach a specific activation condition, or by constructing a trigger circuit that requires multiple excitations of a specific condition. The payload can be designed to achieve a number of malicious outcomes as described in the taxonomy. Certain implementations may only contain a payload (i.e., always-on Trojan). However, such Trojans must not cause any observable impact on the functionality or parametric behavior of the board throughout their activation.

For the addition model, we can generally assume that the PCB design is obtained from a third party vendor and the consumer has no reference design to identify components that are included for malicious purposes. If the design house is trusted and a reference design is available, such Trojans can be still be inserted in a stealthy manner such that they are hard to identify through simple visual inspection. This can be achieved by designing a Trojan with very few components with small footprint and encapsulating components between PCB layers using an "Oreo construction" [7]. However, such insertion currently requires manual intervention of the fabrication process.

In the addition model, we further classify the Trojan designs into two types: 1) Discrete component-based Trojans and 2) IC-based Trojans. Discrete components are single units of circuit components such as transistors, resistors, capacitors, and diodes. On the other hand, ICs contain a combination of discrete elements (e.g., transistors) in a large number and provide more complex functionality within a single component. While an IC could be used to implement a complex Trojan, the presence of an IC that appears unrelated to the intended functionality of the system might raise more suspicion compared to a few discrete components like diodes, resistors, and capacitors spread throughout the board. Therefore, these two models provide a trade-off between functionality and stealthiness. The discrete component-based Trojans in this report are designed by us, while the IC-based Trojans are from known attacks and literature.

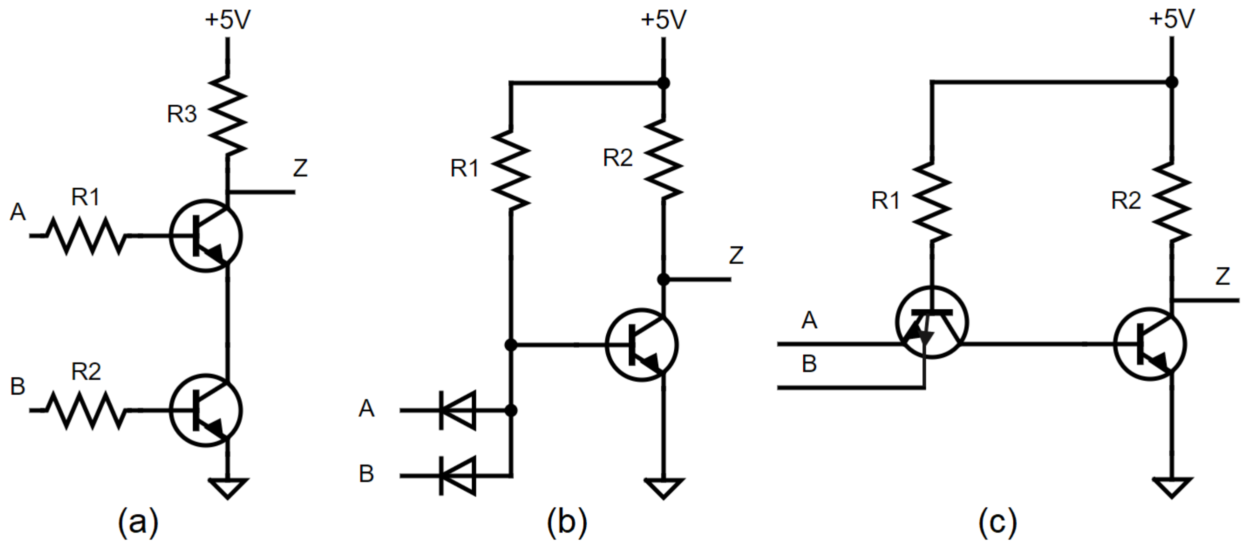


Figure 3. Three different implementation of 2 input NAND: (a) using resistor-transistor logic, (b) diode-transistor logic, and (c) transistor-transistor logic.

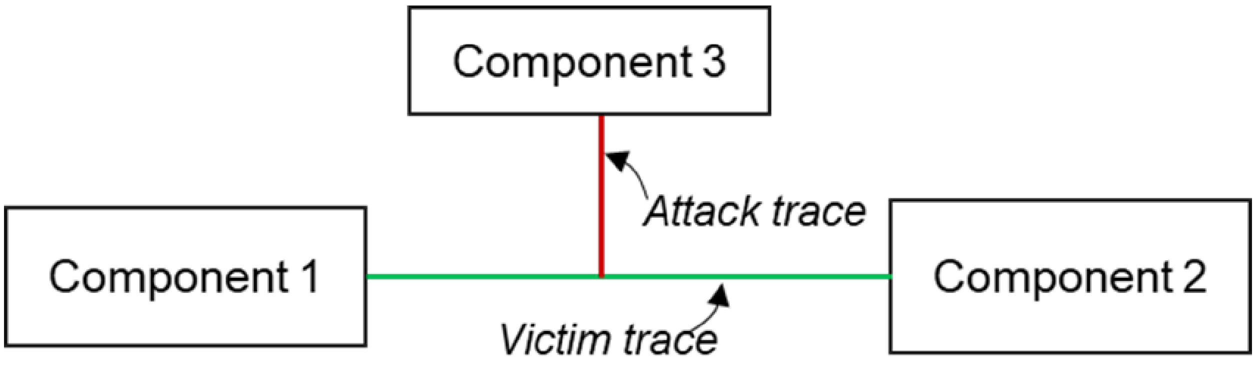


Figure 4. Example of alteration model: signal from component 3 with higher drive strength can overdrive the signal going from component 1 to component 2. Drive strength can be increased through the alteration of trace impedance or structure.

Table 1. Examples of Various Real-World PCB-level Trojan Attacks.

Attack	Primary Modification Type	Location of Modification/Insertion	Attack Goal	Model
Big Hack [8]	Inclusion of small microchip	Between the BMC and FLASH	Alters the BMC firmware	Addition
Modchips of the State [9]	Inclusion of FPGA	Between the BMC and FLASH	Alters the BMC firmware	Addition
Xbox Hack [10]	Inclusion of wires and modchip	LPC Headers and IC	Run unsigned software/games	Addition
Firewalk [11]	RF Transceiver and processor	Dual stacked RJ45 / USB connector.	filtering network traffic	Addition
Cottonmount-I [11]	RF transceiver and digital core	USB	Provide a wireless bridge to network, load exploit software	Addition
Surlyspawn [11]	RF retro-reflector	Data line from keyboard	Broadcasts signals from keyboard	Addition
Trace altration [1]	thinning/rerouting of traces, \\ altering drive strength	JTAG connection and shared bus	signal state hijacking	Addition
Crosstalk Trojan [12]	Alters trace parameters	Traces that may contain critical data	Corrupt data in critical traces	Alteration

4.3.2.2 ALTERATION MODEL

For the alteration model, we explore the various trace modification techniques presented in [1] and [12]. Signal state hijacking techniques have been discussed in [12] through the alteration of drive strengths of different signals connecting together. Such connections are common in JTAG or other debug infrastructures where the debug pins are connected to different pins among several board components.

As shown in Figure 4, trace from component 1 is designed to drive component 3 and component 2. However, by configuring the pin of component 3 as an output with a higher drive strength than the output of component 1, the signal going to the component 2 can be controlled via component 3. Drive strength of a trace can be altered by modifying the trace impedance, length, thickness, or by integrating buffers [12]. Thinning of PCB traces has also been shown in [13] to reduce the lifetime of interconnects (time-triggered functional failure).

The alteration model also includes Trojans that involve the replacement of existing components with recycled, counterfeit, or maliciously altered components. Therefore, new components are not included.

Algorithm 1 Algorithm for Trojan Insertion

```
Procedure: Trojan_Insertion($D_{host}$, $Config$, $Database$)
 1: $G_h$ = constructGraph($D_{host}$)
 2: if $Config.M1 == rand$ then
 3:     $D_{trig}$ = randTemplate($Database.trigger$)
 4:     $D_{pay}$ = randTemplate($Database.payload$)
 5: else if $Config.M1 == user$ then
 6:     $D_{trig}$ = $Config.user\_trigger$
 7:     $D_{pay}$ = $Config.user\_payload$
 8: end if
 9: $G_{trig}$ = constructGraph($D_{trig}$)
10: $G_{pay}$ = constructGraph($D_{pay}$)
11: if $Config.M2 == rand$ then
12:     $N_{wire}$ = countIn($G_{trig}$)                ▷ Count # of input nets
13:     $L_{trig}$ = randNet($G_h$, $N_{wire}$)          ▷ Select $N_{wire}$ nets
14:     $N_{wire}$ = countOut($G_{pay}$)                ▷ Count # of output nets
15:     $L_{pay}$ = randNet($G_h$, $N_{wire}$)           ▷ Select $N_{wire}$ nets
16: else if $Config.M2 == user$ then
17:     $L_{trig}$ = $Config.user\_trigger\_loc$
18:     $L_{pay}$ = $Config.user\_payload\_loc$
19: else if $Config.M2 == rare$ then
20:     $N_{wire}$ = countIn($G_{trig}$)                ▷ Count # of input nets
21:     for each edge $i \in G_h$ do
22:         if $S(i) > Config.S_l \wedge S(i) < Config.S_h$ then
23:             $rareNets$.append($i$)                  ▷ Enumerate rare nets
24:         end if
25:     end for
26:     $L_{trig}$ = randNet($rareNets$, $N_{wire}$)
27:     $N_{wire}$ = countOut($G_{pay}$)
28:     $L_{pay}$ = randNet($G_h$, $N_{wire}$)
29: end if
30: $G_{trojan}$ = mergeTroj($G_h$, $G_{trig}$, $L_{trig}$, $G_{pay}$, $L_{pay}$)
31: $D_{trojan}$ = constructDesign($G_{trojan}$)
32: return $D_{trojan}$
```

Figure 5. Algorithm for Trojan Inertion.

4.4 Toolflow for Synthetic Trojan Benchmark

The proposed toolflow is capable of inserting Trojans presented in the addition model. Both discrete component-based Trojans and IC-based Trojans can be inserted. Since the toolflow uses

KiCAD netlist, the user can include a desired Trojan by drawing the schematic in KiCAD and exporting the netlist.

KiCad can leverage ngspice to perform SPICE based circuit simulation. Therefore, the toolflow offers simulation-based insertion mode, along with random and user-defined insertion. It also provides dynamic alteration of the Trojan during insertion to create variants of the static templates.

The Trojan insertion flow of our tool is presented in Algorithm in Figure 5. Algorithm for Trojan Inertion. The tool inserts a Trojan in a user-provided PCB netlist (D_{host}) according to configuration parameter list *Config* that contains several variables. Using these variables under the *Config* list, the user can define the operational modes such as random or user-defined, desired trigger/payload templates, desired location for insertion, etc. For instance, the mode variable *M1* (in line 2) helps the user to choose between random and user-defined selection of trigger and payload circuits, and mode variable *M2* (in line 11) selects between various insertion modes (i.e., random, user-defined, and switching activity-based). The user also provides a *Database* directory that contains trigger and payload templates in KiCAD netlist format. The final output of the tool is the Trojan inserted PCB netlist D_{Trojan}. Below we describe the individual steps in more detail.

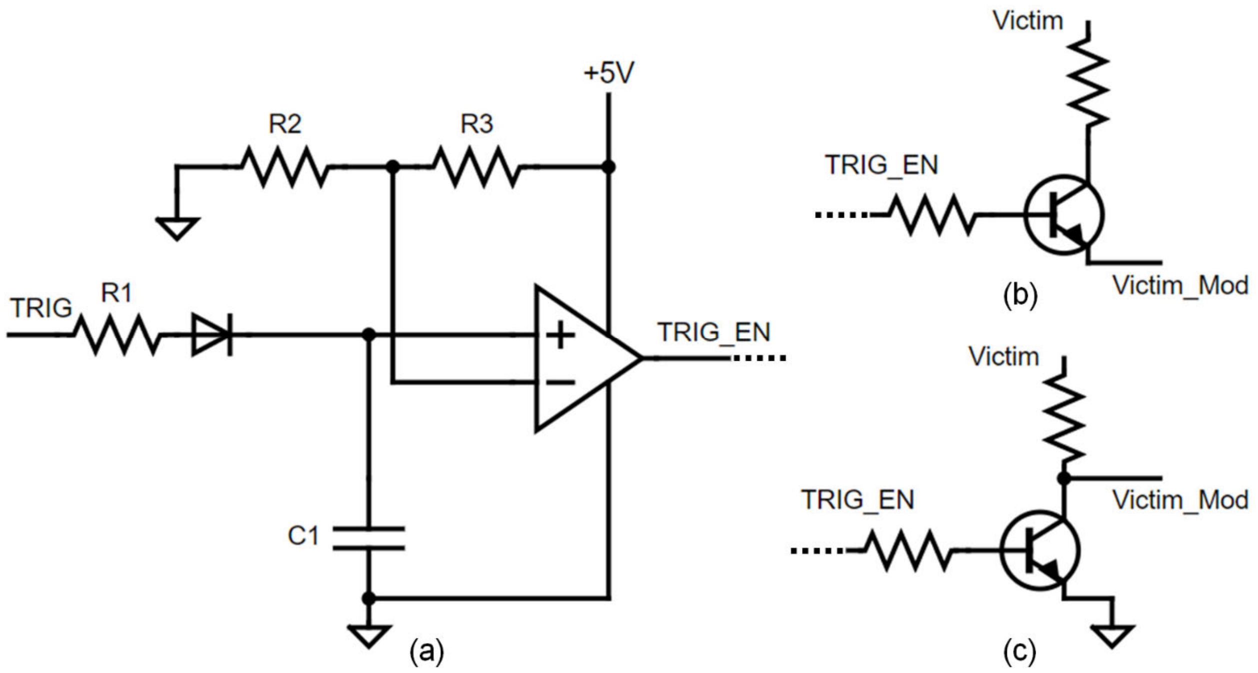


Figure 6. (a) Trigger circuit similar to A2 Trojan [14] where capacitor and op-amps are used to design activation mechanism that requires the rare condition to occur multiple times. Payload with single transistor: (b) impacts the victim when the transistor is off, (c) impacts the victim when the transistor is on.

4.4.1 SELECTION OF TROJAN AND INSERTION LOCATION

As shown in Algorithm in Figure 5. Algorithm for Trojan Inertion. the PCB design provided by the user D_{host} is parsed by the tool using the **constructGraph** function and a bi-directional graph G_{host} is generated. The graph formed is bi-directional as a PCB netlist contains interconnects for which data/current can flow in either direction. The user can select between two different Trojan template modes using *M1*, random and user-defined. The random mode arbitrarily chooses a trigger and payload template from the $Database$ directory using the **randTemplate** function. Two separate database directories for trigger and payload are provided by the user (i.e., *Database.trigger* and *Database.payload*). Currently, the database contains various discrete component-based trigger and payloads. The integration of a template within the database requires the user to modify the trigger and payload net names in the KiCAD netlist to add labels such that the tool can identify the nets to be connected to the host design and the connection type. There are two major types of connections, continuous and discontinuous. Continuous connection involves directly connecting two nets. and, discontinuous connection involves passing a host design's net through a Trojan component. For instance, in Figure 6 (a), the *TRIG* input of the trigger circuit should be directly connected to a net in the host design. To include the label in the trigger netlist, the input net names of the template should be changed to $TRIG_{TriggerCont}$ to indicate a trigger net with the continuous connection.

Conversely, the discontinuous connection is generally used for certain payloads. As shown in Figure 6(b), a victim net in the host design must be broken and connected to *Victim* and $Victim_{Mod}$ such that it passes through the payload transistor. To insert the labels in a netlist, *Victim* and $Victim_{Mod}$ should be converted to *Victim_Payload_Discont_Begin* and *Victim_Mod_Payload_Discont_End* respectively.

Once the modification is done, the netlist is copied to the corresponding database.

The user-defined trigger and payload selection mode is used when the user wants to insert a specific Trojan, instead of randomly selecting from the database. In this mode, the user must provide the file directories of the desired trigger and payload using the *Config* parameter list (lines 6 and 7). To insert IC-based Trojans that combine trigger and payload circuitry in one IC, we provide the IC design as a trigger file and keep the payload directory empty. The tool automatically understands the Trojan model and connects it to the host design according to the labels of trigger and payload nets in the netlist.

Once the trigger and payload templates are selected, their location of insertion in the host PCB design is identified. The user can choose between three different insertion modes using *M2*, random, user-defined, and switching activity-based insertion. For random mode, the tool first analyzes the trigger and payload templates to be inserted and counts the number of nets for which a connection to the host design is required. The tool then identifies all eligible nets in the host design for connecting the trigger and payload wires and then randomly selects from them using the **randNet** function (lines 13 and 15).

The tool contains predefined rules to avoid redundant and invalid connections during the random selection of nets. For instance, GND and VDD nets in the host design are generally not eligible nets for connecting the trigger or payload. The tool also avoids selecting the same nets in the host design for connecting the trigger and payload of the same Trojan as it could create a feedback loop. The tool avoids selecting the same host design net for connecting two different trigger nets as it would increase the activation probability of that Trojan, making it less stealthy. The tool also ensures that multiple Trojans inserted in the same host benchmark will use unique sets of trigger and payload nets to avoid placing the Trojan in the same location.

If *M2* is user-defined, the tool simply takes the insertion location for trigger and payload through a text file containing the location and connection information (i.e., *user_trigger_loc* and *user_payload_loc*). If the user is interested in placing the Trojan in the desired target location, this mode is preferable. In this mode, the tool follows the information obtained from the text file, instead of the labels in the netlist. Each line of the text file should contain the trigger/payload net name, the corresponding host design net name to be connected, and the connection type (continuous/discontinuous). For each discontinuous connection, two trigger/payload net names should be provided that will connect to the two segments of the host design net after it is divided into two nets. Before insertion, the tool will divide the host design net (e.g., N1) and create a new net with prefix Mod (e.g., N1_Mod). Therefore, for discontinuous connection, we include two net names (e.g., Victim and Victim_Mod) for one host design net (e.g., N1). During insertion, Victim connects to N1 and Victim_Mod connects to N1_Mod.

For simulation-based insertion, the tool performs a simulation of the host PCB design using ngspice for large number of cycles and collects the signal transition information to identify rare nets. The simulation data from ngspice is dumped to a text file and the data is analyzed using a Python script to calculate the switching activity of each net (list S in line 22). The user must provide a lower and upper bound of switching activity as input parameter using S_{l} and S_{h} respectively if the simulation-based insertion is selected.

The tool then identifies eligible nets for connecting with the trigger circuit by comparing the switching activity of each net with a user-provided range. Finally, the tool randomly selects from these candidate rare nets using the **randNet** function.

Algorithm 2 Trojan Variant Generation of Discrete Templates

```
Procedure: Variant_Generation($G_{trojan}$)
1:  $L_R$ = find_R($G_{trojan}$)                                  ▷ List of Resistors
2:  $L_C$ = find_C($G_{trojan}$)                                  ▷ List of Capacitors
3:  $numR$ = rand(0, len($L_R$))                                  ▷ # of resistors to alter
4:  $numC$ = rand(0, len($L_C$))                                  ▷ # of capacitors to alter
5:  $R\_Mod\_List$ = randSel($numR, L_R$)
6:  $C\_Mod\_List$ = randSel($numR, L_C$)
7:  for $i \in R\_Mod\_List$ do
8:      $alterMode$ =randSel($series, parallel, percent$)
9:      $R_{new}$ = alterR($i, alterMode$)                        ▷ Alter Resistor
10:     $G_{trojan}$ = modify_graph($G_{trojan}, R_{new}, i$)
11:     if $alterMode == percent$ then
12:         Verify ($G_{trojan}$)
13:     end if
14: end for
15: for $i \in C\_Mod\_List$ do
16:     $alterMode$ = randSel($series, parallel, percent$)
17:     $C_{new}$ = alterC($i, alterMode$)                        ▷ Alter Capacitor
18:     $G_{trojan}$ = modify_graph($G_{trojan}, C_{new}, i$)
19:     if $alterMode == percent$ then
20:         Verify ($G_{trojan}$)
21:     end if
22: end for
23: return $G_{trojan}$
```

Figure 7. Algorithm for Trojan Variant Generation of Discrete Templates

4.4.2 DYNAMIC VARIANT CREATION

While functional diversity can only be achieved by designing new Trojans with different functionality and integrating the insertion process within the tool, structural diversity can be achieved automatically by creating functional variants of the existing templates. The variant of a template should provide the same functional behavior but appears structurally different.

The tool supports the alteration of an existing template prior to insertion to create variants. For certain discrete templates, as shown in Figure 6(a), the tool can automatically alter the resistance and capacitance value within a certain percentage such that the functionality is not changed.

New resistor and capacitor can be included by series to parallel conversion of resistors and capacitor without changing the equivalent resistance/capacitance. The tool contains functions that perform this transformation to the graph representation of the templates prior to insertion. The function can be enabled by the user by setting a flag in the input parameter. The transformations are performed differently in each insertion.

Algorithm in Figure 7. Algorithm for Trojan Variant Generation of Discrete Templates shows the algorithm to generate the variants. This algorithm only executes for the discrete templates. For a given template, it first identifies all resistors and capacitors in the trigger/payload template.

Next, the tool randomly selects a certain number of resistors and capacitors to alter. For each resistor and capacitor, the type of alteration (series-parallel transformation and minor value change) is also randomized. By randomizing the candidate number of elements to alter and their alteration type, a large number of variants can be created from a limited number of hand-crafted Trojan designs. To ensure functional correctness under minor resistance and capacitance value change, verification of the variant is performed (e.g., line 11 and line 19 in Figure 1Figure 7 **Error! Reference source not found.**). In our case, we perform SPICE simulation.}

4.4.3 POST-INSERTION VERIFICATION AND ANALYSIS

The proposed toolflow provides a two-phase verification process to ensure that the Trojan is correctly inserted, and the insertion process has not caused any invalid connection. The first verification phase occurs when the Trojan is inserted by altering the graph D_{host}. The tool checks if the Trojan template graphs (e.g., D_{trig}, D_{pay}) are correctly connected to the desired edges of D_{host}. After insertion, the tool parses the altered graph D_{trojan} and checks if there exists a direct connection between the corresponding edges of D_host and expected D_trig/D_pay nets. If the tool finds no connection or connections with other components in the path, the tool will report the issue which can be checked by visualizing the graph.

Certain invalid connections created during the insertion process will be indicated by the error message when it is opened by KiCAD and an initial layout is generated. As shown in Figure 8, KiCAD automatically renders the connections among the different components and correct rendering of the design requires a design free from invalid connection as shown in Figure 9. Additionally, the user can invoke ngspice to perform a simulation of the Trojan inserted netlist to perform further analysis to confirm the desired malicious functionality.

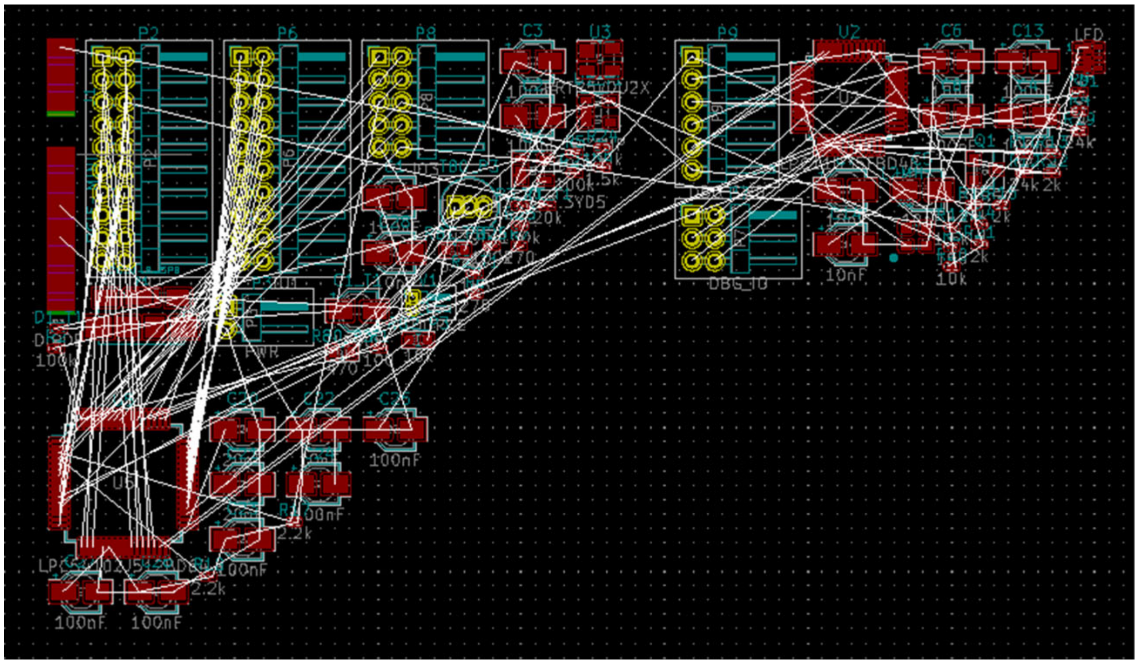
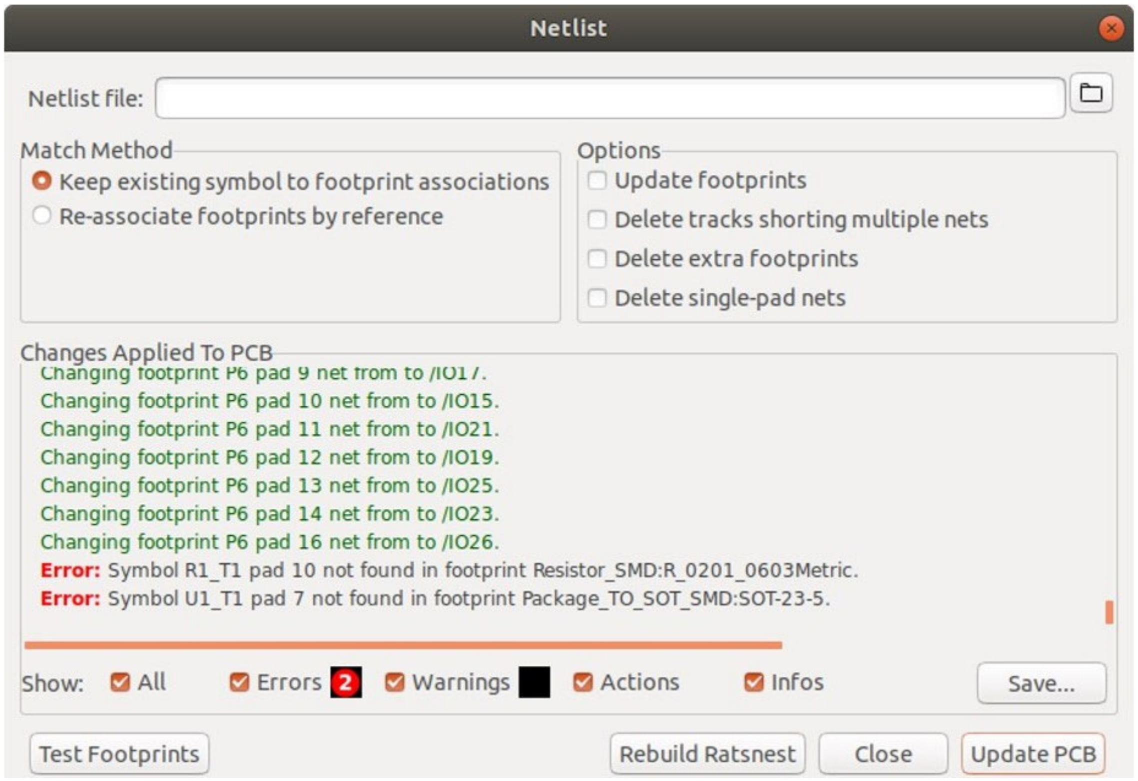


Figure 8. Layout of the Trojan inserted PCB using KiCAD.



Match Method: [✓ Keep existing symbol to footprint associations] Re-associate footprints by reference

```
Changing footprint P6 pad 9 net from to /IO17.
Changing footprint P6 pad 10 net from to /IO15.
Changing footprint P6 pad 11 net from to /IO21.
Changing footprint P6 pad 12 net from to /IO19.
Changing footprint P6 pad 13 net from to /IO25.
Changing footprint P6 pad 14 net from to /IO23.
Changing footprint P6 pad 16 net from to /IO26.
Error: Symbol R1_T1 pad 10 not found in footprint Resistor_SMD:R_0201_0603Metric.
Error: Symbol U1_T1 pad 7 not found in footprint Package_TO_SOT_SMD:SOT-23-5.
```

Errors 2

Figure 9. Snapshot of error message in KiCAD while opening a Trojan inserted netlist with invalid connec-tions.

4.4.4 FABRICATION

While certain countermeasures may only require the Trojan inserted netlist, or its graph representation for analysis, other detection techniques like side-channel analysis may require a fabricated Trojan inserted PCB design for evaluation.

The layout in Figure 8 can be manually routed by the user to generate the final Trojan inserted PCB design that can be fabricated. Alternatively, the Trojan inserted netlist can be directly provided to other PCB design softwares like EasyEDA where the generation of routed design could be even easier [15]. Fig. \ref{fig:layout} (a) shows the routed version of the Trojan inserted PCB design. As shown in Fig. \ref{fig:layout} (b), KiCAD can also generate a 3-D model of the PCB automatically from the netlist where we marked the Trojan components in red. After routing, the PCB design can be easily fabricated by sending the output files from KiCAD to a PCB foundry and assembler. The following output files can be generated by KiCAD for fabrication:

footprint position file, drill file, footprint report, IPC-D-356 netlist, bill of materials, and Gerber files.

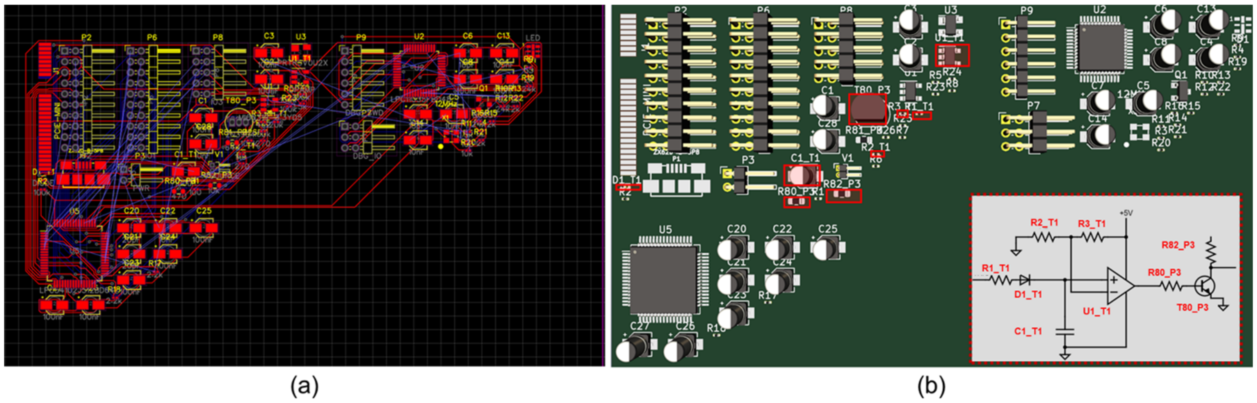


Figure 10. (a) Routed PCB layout obtained from the netlist after Trojan insertion using the tool. (b) 3-D view generated from the same netlist in KiCAD with the Trojan components marked in red.

4.5 RESULTS

In this section, we present analyses related to the Trojan designs, generated benchmarks, and the runtime complexity of the tool. First, we present the side-channel footprint of some of the Trojans using a fabricated Trojan-inserted PCB. Next, we perform a dissimilarity analysis of the benchmarks to show the structural stealthiness of the Trojans and the diversity of the benchmarks we used. Finally, we present a complexity analysis of our tool.

Table 2. Trigger and Payload of the Fabricated Trojans.

Trojan Name	Trigger Type	Payload Type
Trojan1	4 input AND	XOR
Trojan2	4 input NOR	XOR
Trojan3	8 input Mixed	XOR
Trojan4	8 input Mixed with Capacitor	MUX
Trojan5	8 input Mixed with Capacitor	MUX

Table 3. Overhead Analysis for Trojan Inserted PCB Designs

Benchmark	No. of Comp.	No. of Layers	Components	Average Trojan Comp. Overhead (%)	Average Trojan Layout Overhead (%)
CIAA_ACC	569	787	T, D, M, C, R, Mem, SoC	4.1	2.66
EDU_CIAA_K60	187	238	M, C, R, T, D, L, RG	12.5	2.17
CIAA_K60	467	407	D, R, C, M, L, RG, T	5	1.93
CIAA_FSL-MINI	141	188	R, C, M, T, L, D, RG, Mem	16.6	3.21
CIAA-PIC	433	404	R, C, M, T, L, D, RG, Mem	5.4	1.91
EDU-INTEL	87	150	R, C, M, T, L, D, RG	27	3.15
CIAA_PICO	61	144	R, C, M, T, L, D, RG	38.5	13.5
CIAA_SAFETY	275	240	R, C, M, T, L, D, RG, Mem	8.5	2.63
CIAA-Z3R0	33	52	R, C, M, T, L, D, RG	71.2	23.7
Audio_Amp	49	78	R, C, M, T, L, D, RG	117.5	33.06

**M: Microcontroller, T: Transistor, R: Resistor, C: Capacitor, Mem:Memory, L: LED, RG: Regulator, S: SoC, Comp.: Component*

4.5.1 EXPERIMENTAL RESULTS FROM FABRICATED PCB

We designed a PCB Trojan platform by integrating five large discrete component Trojans with different trigger and payload combinations as listed in Table 2. These Trojans are selected due to their large area footprint.

A mix trigger is 4-input AND and NOR triggers combined with one additional transistor. Trojan4 and Trojan5 merges the mix trigger with two different capacitor-based triggers (without and with the diode). We used two 8-bit AVR microcontrollers to drive all the trigger inputs. The XOR payloads corrupt various victim signals going to the seven-segment displays. The MUX payloads were connected to leak victim signals via highly probable or observable points in the PCB, such as IOBs and LEDs. The fabricated PCB is shown in Figure 11.

To analyze the side-channel footprint of each Trojan, we measured the board-level current while keeping the Trojan in two states: inactive and active. In inactive mode, the input to the Trojan under consideration was kept fixed at a value that does not trigger the Trojan. The rest of the PCB that includes the other four Trojans were applied random vectors from the two microcontrollers. In active mode, the microcontrollers caused switching activity in the Trojan under consideration by applying vectors that alternatively activated and deactivated the Trojan in quick intervals. The activity of the rest of the PCB was kept the same as in the inactive mode. The current consumption of the five Trojans in active and inactive states is presented in Figure 12. We can observe a maximum of 3.5% increase in current consumption (for Trojan5). Without any reference design to obtain a golden signature (assuming untrusted design house), such a minor increase in current consumption is unlikely to raise any suspicion. In the presence of golden design, smaller Trojans can be inserted between the layers.

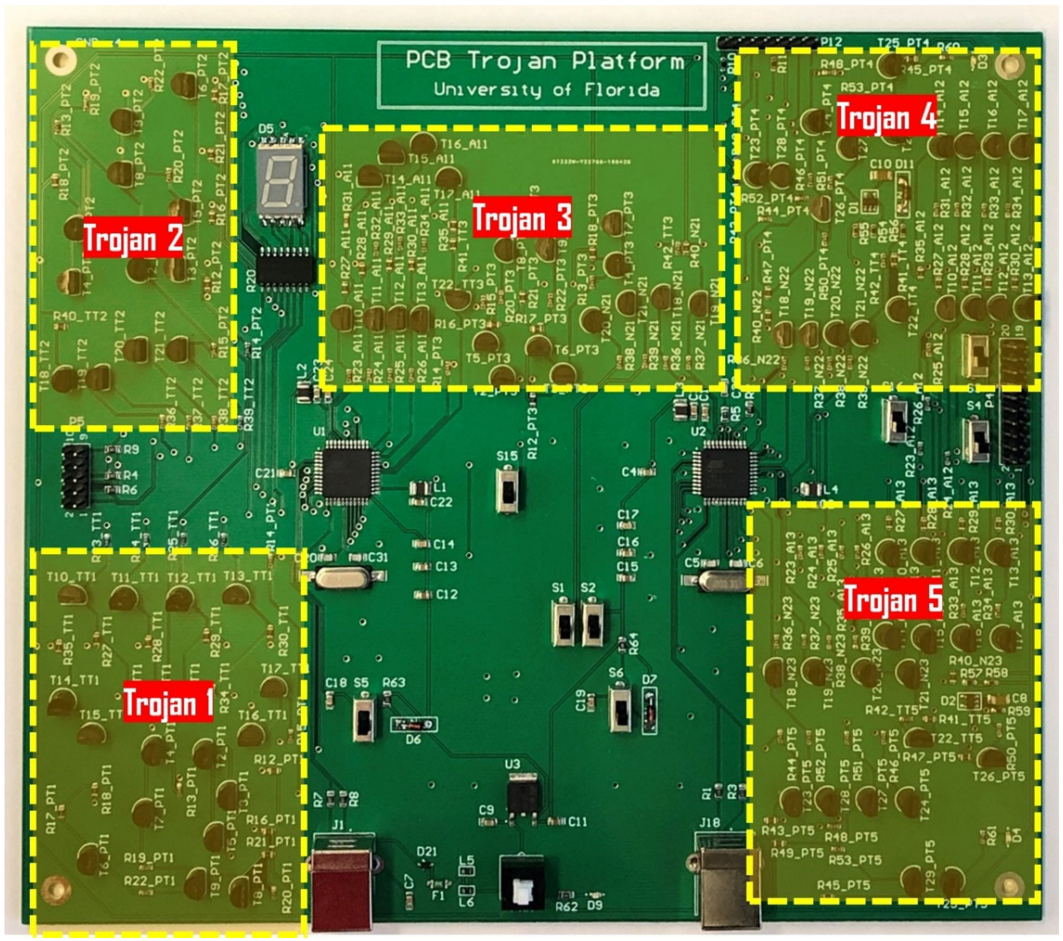


Figure 11. Fabricated PCB Trojan platform designed from our benchmarks.

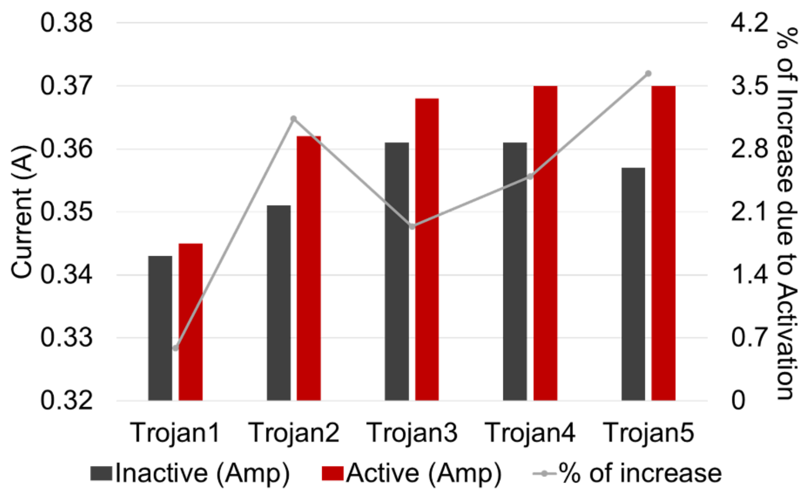


Figure 12. Increase in current consumption due to Trojan activation (obtained from fabricated PCB).

4.5.2 PCB TROJAN BENCHMARK GENERATION

Using the 10 sample PCB designs presented in the first column of Table 3, we generated large number of Trojan benchmarks.

Most of the sample designs described are part of the CIAA (Computadora Industrial Abierta Argentina) project which is an open-source repository of various PCB designs [16], [17]. The sample designs are of varied complexities in terms of the number of components, number of connections, and types of components. The designs vary from containing simple components like resistors, capacitors, LEDs, transistors, etc. to complex components like SoC, microcontroller, memory, etc.

We use different trigger and payload structures described in the addition model and inserted 50 different Trojans in each of the sample designs using the random mode for both M1 and M2. We performed post-insertion verification of the Trojans using the graph-based automatic checking and all 500 Trojan inserted benchmarks were found to have the desired connectivity between the Trojan and the host design. Thus, the proposed toolflow can handle PCB designs with varied complexity.

4.5.3 BENCHMARK DIVERSITY AND TROJAN STEALTHINESS

We evaluated the structural dissimilarity of the golden (i.e., host design) and Trojan inserted benchmarks using a size-independent network similarity evaluation method called Netsimilie [18]. Given two networks or graphs, Netsimilie extracts features of each node of a graph, aggregates the features, and generates a signature vector for each graph. The Canberra distance between the signature vector of the graphs provides the dissimilarity value between the graphs. Lower Canberra distance indicates a lower dissimilarity between the graphs. Higher Canberra distance indicates that the graphs are highly dissimilar. A low Canberra distance between a golden design and a Trojan inserted design indicates the Trojan is structurally stealthy.

The structural features extracted by Netsimilie for each node are based on the local and egonet based features. The egonet of a node is the induced subgraph of the node's neighboring nodes.

The structural features extracted are:

- Number of neighbors of the node
- Clustering coefficient of the node, defined as the number of triangles connected to the node, divided by the number of connected triples centered on the node
- Average number of neighbors that are two levels away from the node
- Average clustering coefficient
- Average number of nodes in the node's egonet
- Number of nodes in the egonet of the node

The above features form a feature matrix of a node. After extracting features from all the nodes of the graph, Netsimilie generates a *node * feature* matrix. The *node * feature* matrix is aggregated to a signature vector by using a set of aggregators on each feature. The aggregators used by Netsimilie are mean, kurtosis, standard deviation, skewness, and median. For each graph, a signature vector is generated. To calculate the distance between the signature vectors, the Canberra distance is used.

To perform this analysis, the 10 PCB designs in Table 3 were inserted with the trigger circuit in Figure 6. The impact of this Trojan on the structure of each golden design as well of as the diversity among the 10 golden and 10 Trojan inserted designs were observed using the Canberra distance. The Canberra distance between all pairs of golden and Trojan inserted designs was calculated and plotted as a heatmap as shown in Figure 13. The rows of the heatmap were normalized with the highest value in that row. The dissimilarity between a golden design and the Trojan inserted design is very low (observable by the dark shade in the heatmap). Therefore, the Trojans inserted in these designs caused very minute structural changes.

The dissimilarity between a pair of different golden designs is very high. Thus, the PCB designs selected for creating the benchmark suite are structurally diverse, making them suitable for extensive and unbiased verification of countermeasure and creation of training database.

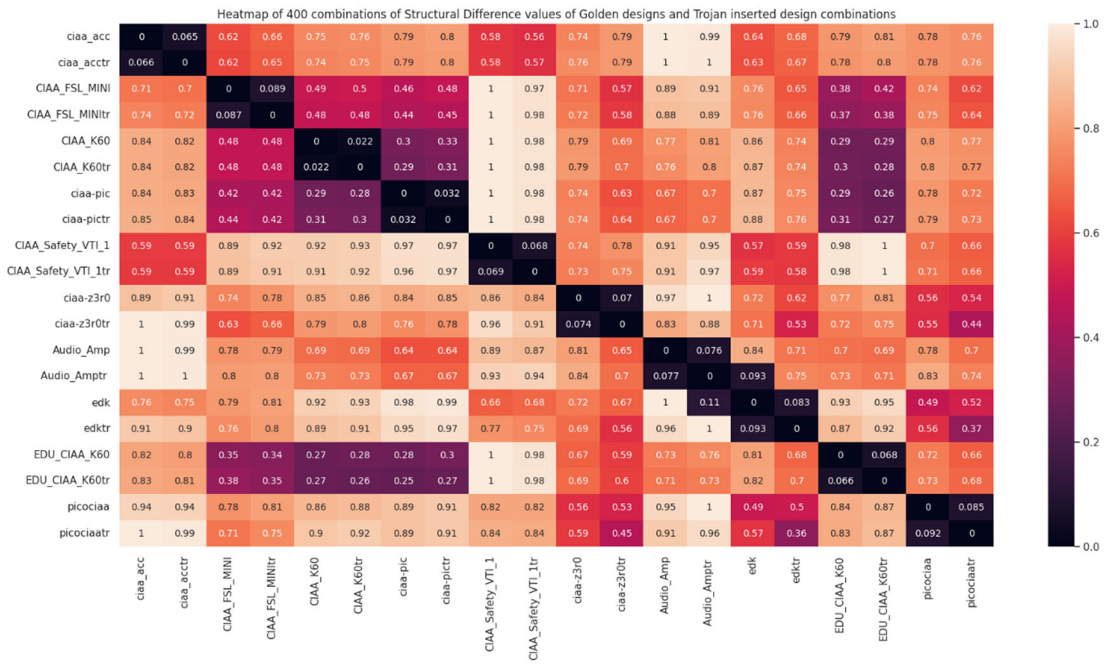


Figure 13. Figure shows the difference metric for different various pairs of PCB designs. Dark shade indicates lower difference and lighter shades indicate higher difference among two designs. The difference between the Trojan free and Trojan inserted version of the same design (e.g., ciaa_acc vs ciaa_acctr) have very minute difference. Conversely, two different benchmarks have very high difference value (e.g., ciaa_acc vs CIAA_K60).

4.5.4 COMPLEXITY ANALYSIS OF THE TOOL

The runtime of the tool primarily depends on the size of the design. The tool first converts the netlist to a graph using the constructGraph function that requires the tool to iterate through each component of the design and register them as an individual node and each net of a component as an edge. Constructing the graph has a complexity of O(n+w), where *n* is the number of components in the design and *w* is the number of nets. Selecting candidate nets for connecting to the trigger and payload circuit is O(rXw), where *r* is the number of triggers/payload nets required and w is the number of nets. Simulation is another significant task that can be performed. Time to perform the simulation is also dependent on the design size and the time to analyze the simulation data depends on the number of nets in the design and the total time duration of the simulation. Simulation is not a required component of our framework and is removed from the calculation. Therefore, the complexity is defined as O(n+w+rXw). Because *r* << *w*, the complexity simplifies to O(n+2w). Furthermore, in designs with complex ICs, the number of nets per component is very high, making *2w* a more impactful component of runtime compared to *n*. We ran the tool on a desktop computer with Intel i7 processor and 8GB RAM. The runtime is presented in Figure 14. For each host design with varied number of nets, the tool inserted 55 different Trojans. The light blue area plot shows the 50 runtime values for each host design. As we can see, the runtime does not vary significantly while inserting different Trojans in the same design and changes proportionally with the number of nets in the host design. For a design with around 800 nets (i.e., ciaa_acc) the runtime is around 1.8 seconds, which shows the scalability of the tool as the PCB design size increases.

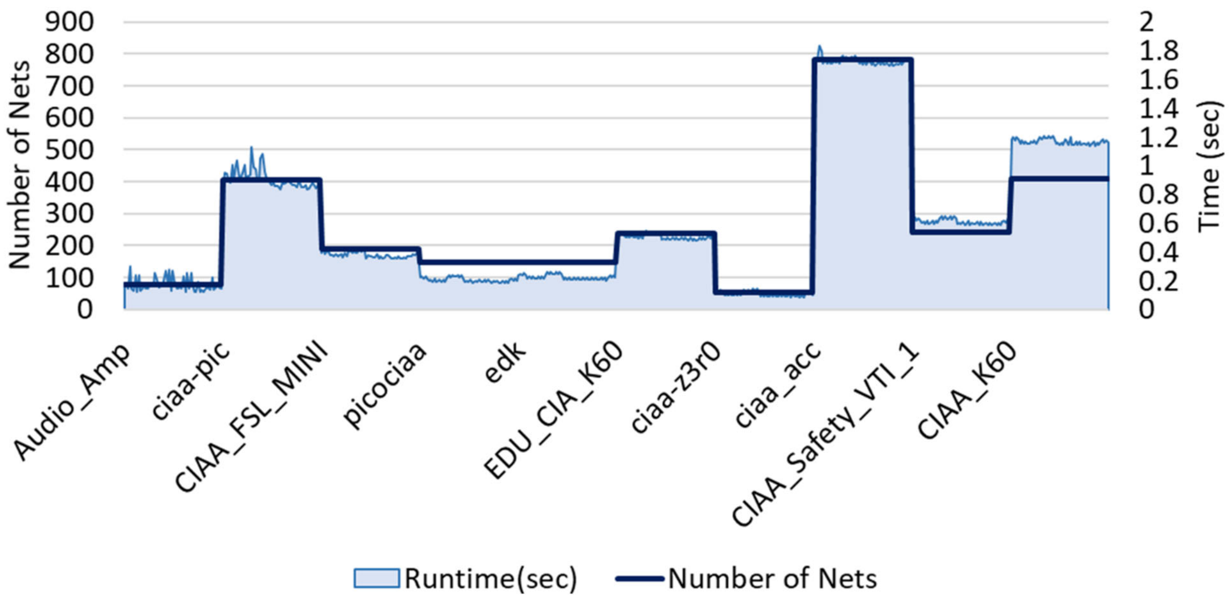
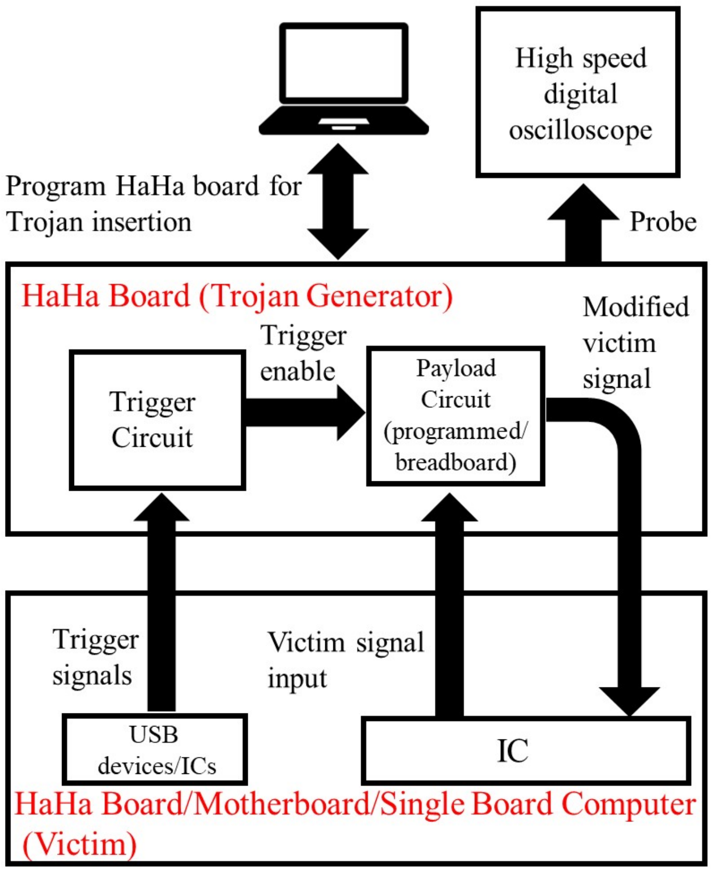


Figure 14. Graph of tool runtime for inserting 50 Trojan instances in host benchmarks of various sizes. The runtime is proportional to the number of nets in the host design. For inserting different Trojans in the same host design, there are minute variations in runtime.

5.0 AN AUTOMATED FRAMEWORK FOR PCB-LEVEL TROJAN BENCHMARKING

5.1 PCB-LEVEL TROJAN INSERTION EMULATOR



High speed digital oscilloscope
Program HaHa board for Trojan insertion
Probe
HaHa Board (Trojan Generator)
Modified victim signal
Trigger enable
Trigger Circuit
Payload Circuit (programmed/ breadboard)
Trigger signals
Victim signal input
USB devices/ICs
IC
HaHa Board/Motherboard/Single Board Computer (Victim)

Figure 15. A high-level view of the system architecture used for Trojan insertion experiments

The architecture of our system for Trojan insertion emulation experiments is depicted in Figure 15. By employing this system, necessary experiments can be conducted using either a custom HaHa board, a computer motherboard, or a single-board computer. It uses one PCB as a victim (into which the Trojans are inserted) and a custom HaHa board as a Trojan generator. The latter uses an on-board Field-Programmable Gate Array (FPGA) to implement the payload circuitry for a functional Trojan. The trigger input sources of each Trojan are derived from ICs on the victim board or peripheral devices. The output of the trigger circuit transmits a "Trigger Enable" signal that can activate each Trojan via its own FPGA-based payload circuit; the latter can utilize input signals from other electronic components and/or devices of the victim PCB. Through the Trojan payload circuit, both communication or side channel information leakage and Trojans of the malicious memory/data write type can be implemented on the victim nodes. Their impacts can include automatic overriding of the state of on-board LEDs, corruption of the readings of sensor ICs or peripheral device parameters with false values, freezing or shutdown of the entire system, or unauthorized readout of secret data bus information from the victim PCB, peripheral device, and/or other electronic components. A high-speed digital oscilloscope was used to probe the modified and/or leakage signals of the victim component or device.

5.2 PCB-level Trojan Models on the Custom HaHa Board

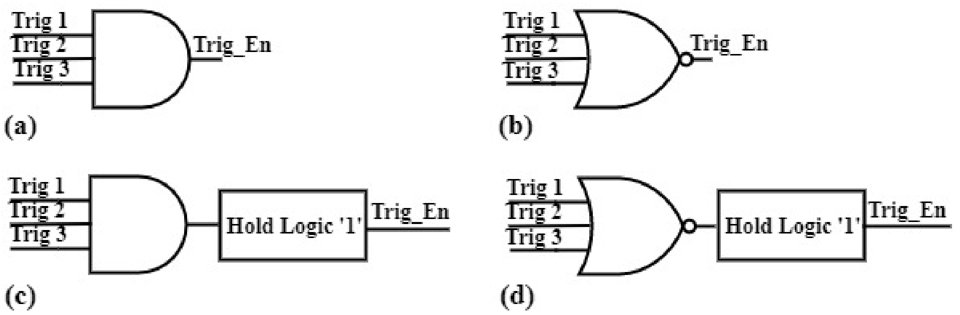


Figure 16. Triggers used for emulating PCB-level Trojan insertion using the custom HaHa board: (a) an AND gate, and (b) a NOR gate. Triggers used for emulating PCB-level Trojan insertion using commercial computer motherboards: (c) an AND gate plus a circuit to hold.

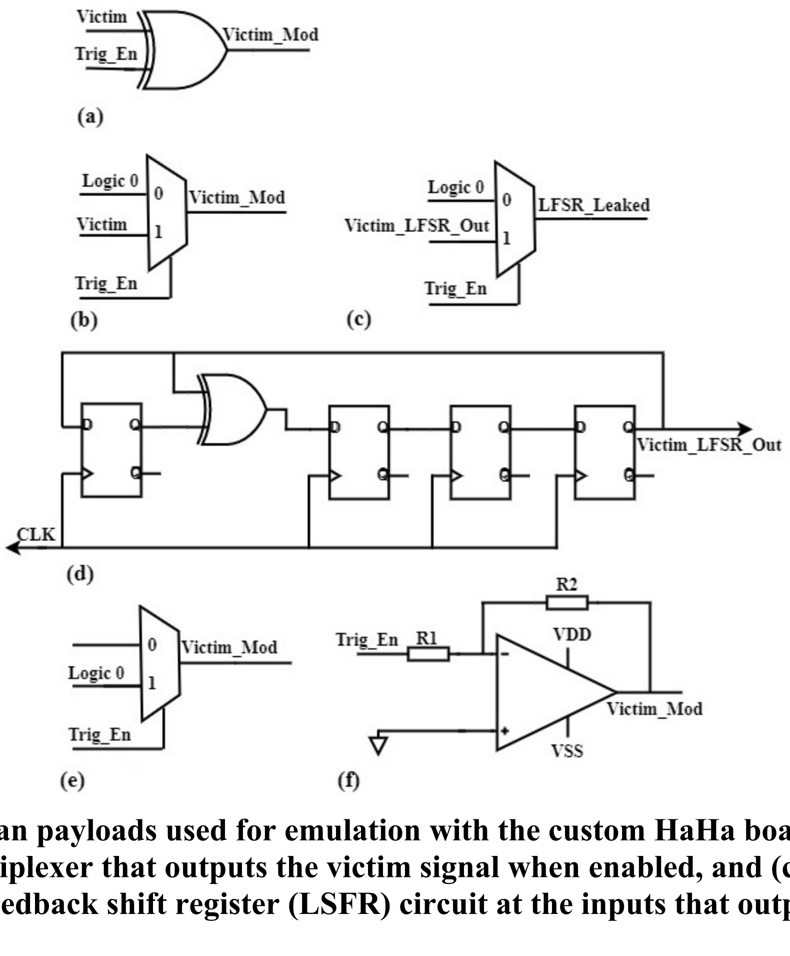


Figure 17. Trojan payloads used for emulation with the custom HaHa board: (a) an XOR gate, (b) a multiplexer that outputs the victim signal when enabled, and (c) a multiplexer with a linear feedback shift register (LSFR) circuit at the inputs that outputs the victim.

Figure 16 and Figure 17 list 2 different types of Trojan trigger circuits (Figure 16 (a)-(b)) and 3 different Trojan payload circuits (Figure 17 (a)-(d)) that were used to generate hardware Trojans using the custom HaHa board. The trigger circuits consist of an AND gate or a NOR gate. The AND gate trigger is enabled only when all its inputs become logic high. By contrast, the NOR gate trigger is enabled only when all inputs are logic low. Both circuits were used to trigger the activation of the Trojan payloads shown in Figure 17. The number of trigger inputs determines the probability of the Trojan being activated. These inputs (denoted by Trig_1, Trig_2 and Trig_3) are derived from a second HaHa board that serves as the victim, as shown in the system architecture diagram (Figure 15).

Three different payload circuits, as shown in Figure 17 (a)-(c), were designed to perform the functions of three types of Trojans. Each Trojan is only activated when its trigger conditions are satisfied. The circuit in Figure 17 (a) uses an XOR gate to invert each input data bit, thus implementing a Trojan that operates as a malicious memory/data write element. Another type of Trojan uses a multiplexer as the payload circuit (as shown in Figure 17 (b)) to leak either communication or side-channel data. This type of Trojan can be made less detectable by randomizing the input information with a linear feedback shift register (LFSR) circuit, as shown in Figure 17 (c)-(d). Different combinations of these trigger and payload circuits can be used to design more sophisticated Trojans that maliciously modify data bus memory and/or automatically leak data bus information on the victim board based on the trigger inputs.

5.3 PCB-LEVEL TROJAN MODELS ON COMMERCIAL DESKTOP MOTHERBOARDS

Trigger circuits for commercial desktop motherboards were designed by adding an extra circuit that holds the trigger output value as long as the trigger logic generates a logic high. Two such Trojan trigger circuits are depicted in Figure 16(c)-(d). The extra circuit acts as a state machine that enters and holds the Trojan activation state as the trigger enable signal Trig_En goes logic high. The purpose of the state machine is to ensure that the Trojan remains activated even when any of the trigger inputs changes from logic '1' back to logic '0'. This is very common when sourcing high-frequency trigger inputs, e.g., from signals on USB data buses or other high-speed input/output ports.

Figure 17 (b) and Figure 17 (e)-(f) depict three payload circuits designed to insert Trojans into commercial PCBs such as motherboards and single board computers. Figure 17 (b) is used for information leakage from any victim IC on a commercial PCB or peripheral device, and is similar to that implemented on the HaHa board. The victim IC can be any chip that transmits and/or receives data - either from other ICs on the board or from peripheral devices. Fig. 8(e) is a circuit designed to maliciously write a logic '0' into a vulnerable node of a peripheral device or an on-board IC when the trigger is enabled (i.e., Trig_En goes logic high). Finally, Figure 17 (f) uses an inverting amplifier circuit to convert positive analog voltages at the input of an on-board analog to digital converter (ADC) into negative voltages, thus maliciously modifying the digitized data. As a result, the direct impact of the two latter Trojans is performance degradation of the victim device and IC. Further impacts, such as a system-wide shut down, may occur if the device or IC is part of a critical on-board component such as the memory controller.

5.4 EXPERIMENTAL SETUP AND DEMONSTRATION

5.4.1 PCB TROJAN IMPLEMENTATIONS ON CUSTOM HAHA BOARDS

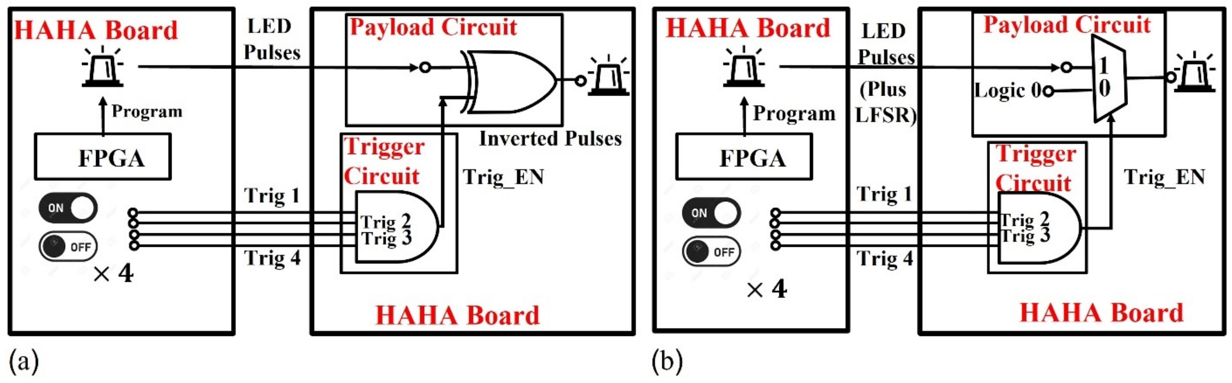


Figure 18. Block diagram of Trojans implemented on custom Haha boards that (a) invert the state of an on-board LED, and (b) leak the state of the LED. In both cases, the Trojan is triggered when the user turns on all four on-board switches.

Several Trojans were designed to perform either malicious data writes on a HaHa board or information leakage operations using an on-board LED. Activation of these Trojans is determined by the state of four user-controlled switches on the victim HaHa board. Fig. 18(a) shows the block diagram of a Trojan inserted in the victim board that is activated when all four switches are turned on; the corresponding payload is maliciously inversion of the LED state. Fig. 18 (b) shows the block diagram of another Trojan that has the same trigger but a different payload, namely leakage of information on the LED state. The leaked data can be masked by adding an LFSR circuit that generates a pseudo-random code. Also, note that the AND-based trigger circuit of both these Trojans can be easily replaced by a NOR-based circuit such that the Trojans are activated only when all the switches turn off.

5.4.2 PCB TROJAN IMPLEMENTATIONS ON COMMERCIAL BOARDS

In order to emulate the insertion of Trojans on commercial boards, we developed four different types of PCB Trojans and implemented them on either a motherboard or a single-board computer. Block diagrams of the first two designs, both of which contain a malicious data write payload, are shown in Fig. 19 and Fig. 20. As in Fig. 6, the experimental system uses a custom HaHa board as the Trojan generator and a desktop computer motherboard or a single-board computer as the victim board whose data buses are explored and modified. The first Trojan (Fig. 19) was implemented on a multi-protocol I/O and monitoring chip (F71889A from Fintek). We exploited a vulnerability in this IC's temperature monitoring function to generate false system temperature readings that can trigger unexpected system shutdowns. Specifically, the payload relies on the fact that the on-board firmware automatically freezes or shuts down the system whenever the temperature sensed by the monitoring IC rises over a certain threshold. For this purpose, the chip uses an internal ADC to digitize the analog output voltage of a board-mounted temperature sensor.

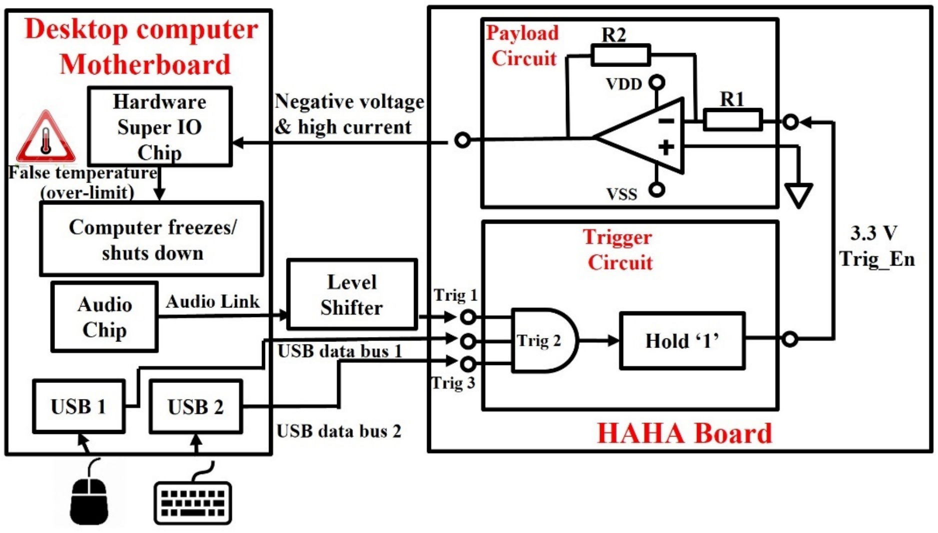


Figure 19. Block diagram of an "automatic computer freeze" Trojan triggered by a combination of USB device plug-in and audio playback events.

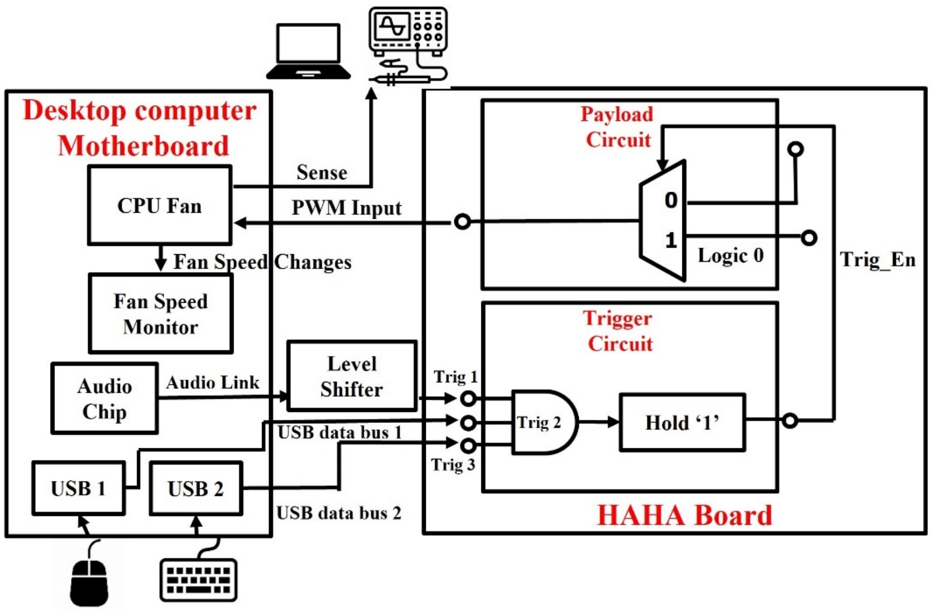


Figure 20. Block diagram of an "automatic CPU fan speed change" Trojan triggered by a combination of USB device plug-in and audio playback events.

The temperature sensor voltage was found to decrease to a negative value when the temperature crosses the threshold value. Accordingly, the payload circuit uses an operational amplifier (op-amp), as shown in Fig. 8(f), to convert the 3.3 V trigger enable signal (Trig_En) to a negative voltage that emulates an above-threshold temperature reading. The analog input pin for the F71889A's temperature monitoring circuit requires a drive current of at least 0.1 A, so a high-output-current op-amp (OPA551PA, from Texas Instruments) was used to implement the payload. The trigger circuit used by this Trojan consists of a three-input AND gate followed by a simple state machine. The latter captures transient logic "high" events at the AND gate output and holds (i.e., latches) them to generate a digital signal (denoted by Trig_En) that permanently triggers the payload. Here, the trigger inputs are two USB devices (USB mouse and keyboard) signals and one audio output signal from an audio codec IC (ALC892, from Realtek) on the motherboard. Accord-ingly, the Trojan is triggered whenever the two USB devices are plugged into specific USB ports (located on the motherboard's rear panel) and an audio file is played for more than 3 sec. The minimum audio duration for triggering the Trojan (3 sec in this case) was optimized to eliminate the possibility of unwanted triggering due to auditory noise (e.g., short-duration sounds generated when USB devices are plugged in). In addition, a level shifter IC is used to convert the low-voltage (~1 V) audio signals to 3.3 V logic signals that can be directly read by the FPGA.

The second Trojan was designed to maliciously modify CPU fan operation, as shown in Fig. 20. The victim node is the pulse-width modulation (PWM) input pin of the four-wire CPU fan on the motherboard, which can be attacked by overriding the PWM signal with a logic '0'. This results in a change of CPU fan speed, which can be confirmed through either a hardware monitoring program or directly via the sense pin of the CPU fan connector. The payload for this Trojan is identical to that shown in Fig. 8(e). Specifically, the multiplexer only passes a logic '0' to the PWM pin when the trigger enable signal (denoted by Trig_En) is detected as a high level. The trigger circuit uses a two-input AND gate with a state machine, as shown in Fig. 7 (c). Here, the inputs are bus signals of two USB devices (USB mouse and keyboard). Thus, the impact of this Trojan is automatic and malicious change of CPU fan speed when both USB devices are plugged in.

The third and fourth Trojans (shown in Fig. 21 and Fig. 22, respectively) are both focused on information leakage. The first design uses a AND-gate trigger and a multiplexer (as in Fig. 8(b)) as the payload circuit. Its goal is to leak some data from the audio bus whenever the trigger is enabled, i.e., both USB devices are physically connected. A level shifter was again used to convert low-voltage audio signals to 3.3 V. The second Trojan design (Fig. 22) leaks data bus information from an Ethernet controller IC (RTL8152B, from Realtek) on a single-board computer through a physical-layer (PHY) transmit pin in 10 Mbps mode. The RTL8152B is used for high-speed wired networking over CAT-5 UTP cable (100 Mbps) or CAT-3 UTP cable (10 Mbps).

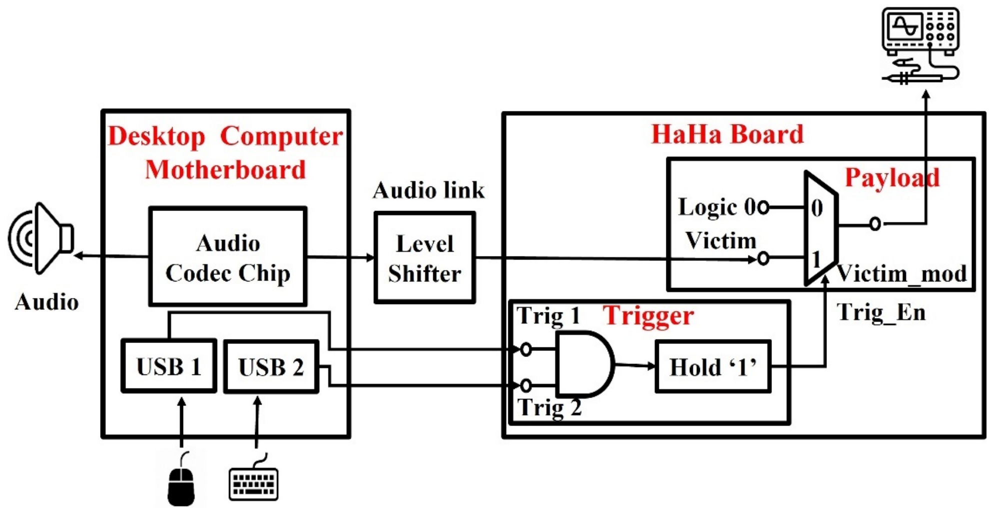


Figure 21. Block diagram of an "automatic audio data leakage" Trojan triggered by the plug-in of USB devices.

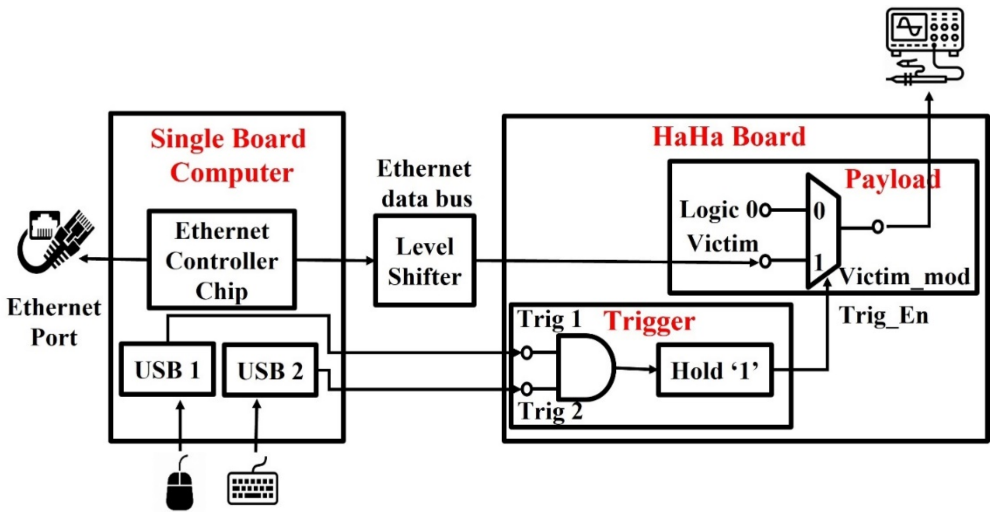


Figure 22. Block diagram of an "automatic networking data leakage" Trojan triggered by the plug-in of USB devices.

5.5 SETUP FOR EXPERIMENTAL MEASUREMENTS

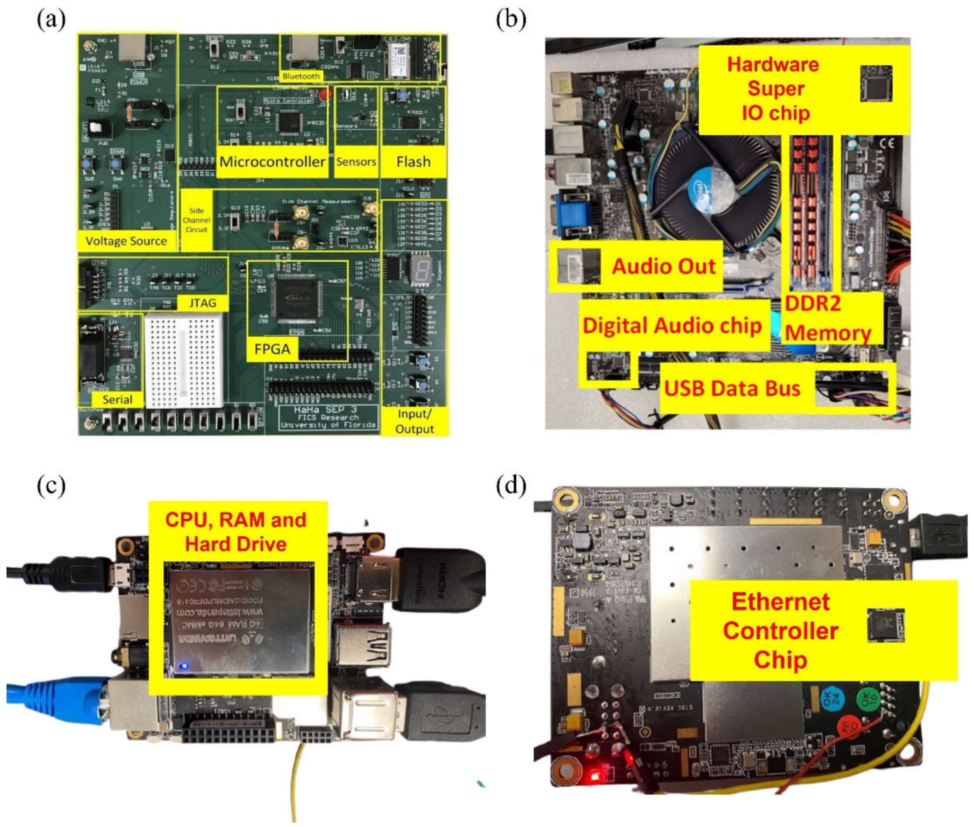


Figure 23. (a) The custom hardware security platform (HaHa board) used for building the trigger and payload circuits. (b)-(d) Trojan insertion victims: (b) a motherboard, (c)-(d) a single-board computer (both front and back sides shown).

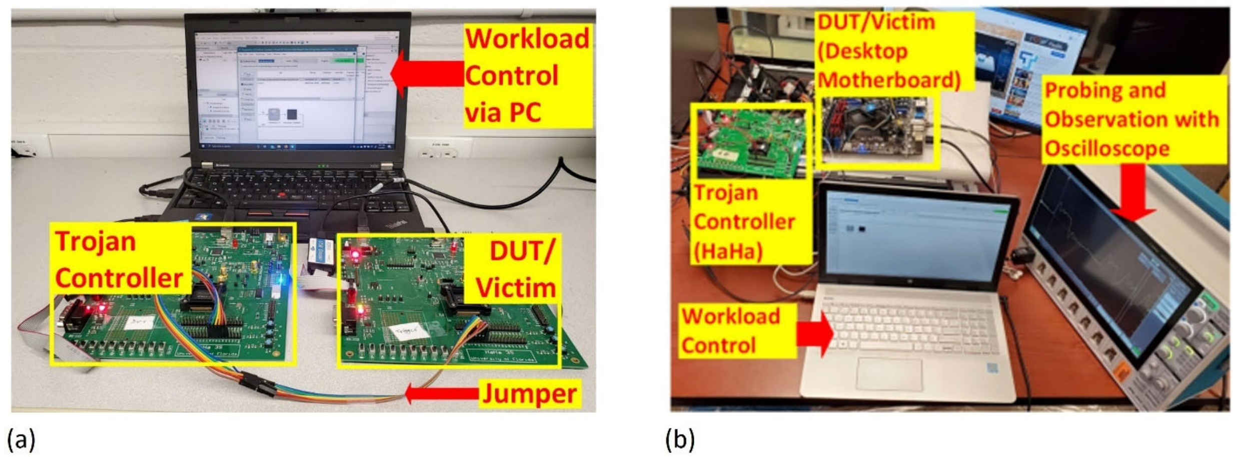


Figure 24. Experimental setup for emulating Trojan insertion using (a) the HaHa board, and (b) the motherboard.

A custom HaHa board (shown in Fig. 23(a)) was used to implement Trojan generation for the proposed Trojan emulation architecture (shown in Fig. 6). The board contains an ARM-based microcontroller, a low-power FPGA (Intel MAX-10 with 50K logic elements), a flash memory module, a side channel measurement module, an inertial measurement unit (IMU), a breadboard, and a few other basic elements commonly found on commercial FPGA development boards (including a seven-segment display, I/O modules, serial ports, switches, and buttons). The two primary modules used in the experiments were the FPGA chip and the breadboard; the former was programmed to implement the digital components of both trigger and payload circuits, while the latter was used to implement the analog components. The side-channel measurement module uses an instrumentation amplifier that is configured to measure the current flowing through a 1 Ω current-sensing resistor.

The desktop motherboard (MSI H67MA-E45-B3d with 16 GB DDR3 RAM) that was used as a Trojan insertion victim is shown in Fig. 23(b). The other victim used during the experiments was a single-board computer (LattePanda) with 4 GB RAM and 64 GB eMMC storage; its front and back sides are shown in Figs. 23(c)-(d). These two boards were chosen as being representative of older desktop computers and modern miniaturized computing platforms, respectively.

Fig. 24(a)-(b) depict the setup used for Trojan insertion experiments on the HaHa board and the motherboard, respectively. The experimental setup using the single board computer as the victim is exactly the same as for the motherboard. A second HaHa board serves as the Trojan generator if a HaHa board is chosen as the victim, as shown in Fig. 24(a). The same HaHa board also generates Trojans when the motherboard or single-board computer is chosen as the victim, as shown in Fig. 24(b). In both cases, a high-speed digital oscilloscope (1 GHz bandwidth) was used to probe waveforms on the victim board. Finally, a laptop computer was used to compile the digital Trojan circuits (defined by Verilog netlists) and then program the FPGA on each HaHa board.

5.6 TROJAN ACTIVATION AND VALIDATION RESULTS

The setups shown in Fig. 24 were used to perform a variety of Trojan insertion experiments on both custom HaHa boards and commercial boards (motherboards and single-board computers). The results demonstrate successful Trojan insertion and payload activation, as described next.

5.6.1 HARDWARE TROJAN INSERTION RESULTS ON THE HAHA BOARD

We first conducted experiments to verify the effectiveness of all five Trojans designed using the HaHa board as a victim board, as listed in Table 4. The victim node explored during the experiments is an LED that is programmed by the workload laptop to toggle every 0.76 sec. The Trojan trigger inputs are four switches on the victim HaHa board. In each case, the final experimental results matched the desired impacts of the Trojan payload. Trojans 1 and 2 were activated when all their trigger inputs became high or low, respectively, resulting in inversion of the LED state. Trojans 3 and 4 were triggered by the same inputs to leak state information from the LED on the victim board to another LED on the Trojan generator. Trojan 5 was also verified to function as expected; the LED state information received by the Trojan generator was equivalent to the randomized information from the victim board. Table 4 lists the power consumption of the victim board (i.e., the device under test (DUT)), in two scenarios: i) when the Trojans are not activated, and ii) when they are activated. Among the five designs, Trojans 1, 2, and 5 result in a significant increase in power consumption when they are activated. As a result, these Trojans, which initiate either malicious data writes or masked information leakage, are relatively easy to detect using on-board supply current monitoring.

Table 4. Summary of Trojan designs implemented on the HaHa board

Trojan ID	Trigger Input	Trigger Circuit	Payload	Payload circuit	DUT current (mA), Trojan not activated	DUT current (mA), Trojan activated
Trojan 1	4 switches	4-input AND gate	Invert the state of one LED on the victim board	2-input XOR Gate	14.01	36.44
Trojan 2	4 switches	4-input NOR gate	Invert the state of one LED on the victim board	2-input XOR Gate	5.23	17.86
Trojan 3	4 switches	4-input AND gate	Leak the state of one LED on the victim board to an LED on the Trojan generator	2-input MUX	4.84	5.73
Trojan 4	4 switches	4-input NOR gate	Leak the state of one LED on the victim board to an LED on the Trojan generator	2-input MUX	4.22	5.2
Trojan 5	4 switches	4-input AND gate	Leak the randomized state of one LED on the victim board to an LED on the Trojan generator	2-input MUX	21.56	26.82

5.6.2 HARDWARE TROJAN INSERTION RESULTS ON THE HAHA BOARD

We first conducted experiments to verify the effectiveness of all five Trojans designed using the HaHa board as a victim board, as listed in Table 4. The victim node explored during the experiments is an LED that is programmed by the workload laptop to toggle every 0.76 sec. The Trojan trigger inputs are four switches on the victim HaHa board. In each case, the final experimental results matched the desired impacts of the Trojan payload. Trojans 1 and 2 were activated when all their trigger inputs became high or low, respectively, resulting in inversion of the LED state. Trojans 3 and 4 were triggered by the same inputs to leak state information from the LED on the victim board to another LED on the Trojan generator. Trojan 5 was also verified to function as expected; the LED state information received by the Trojan generator was equivalent to the randomized information from the victim board. Table 4 lists the power consumption of the victim board (i.e., the device under test (DUT)), in two scenarios: i) when the Trojans are not activated, and ii) when they are activated. Among the five designs, Trojans 1, 2, and 5 result in a significant increase in power consumption when they are activated. As a result, these Trojans, which initiate either malicious data writes or masked information leakage, are relatively easy to detect using on-board supply current monitoring.

5.6.3 HARDWARE TROJAN INSERTION RESULTS ON COMMERCIAL BOARDS

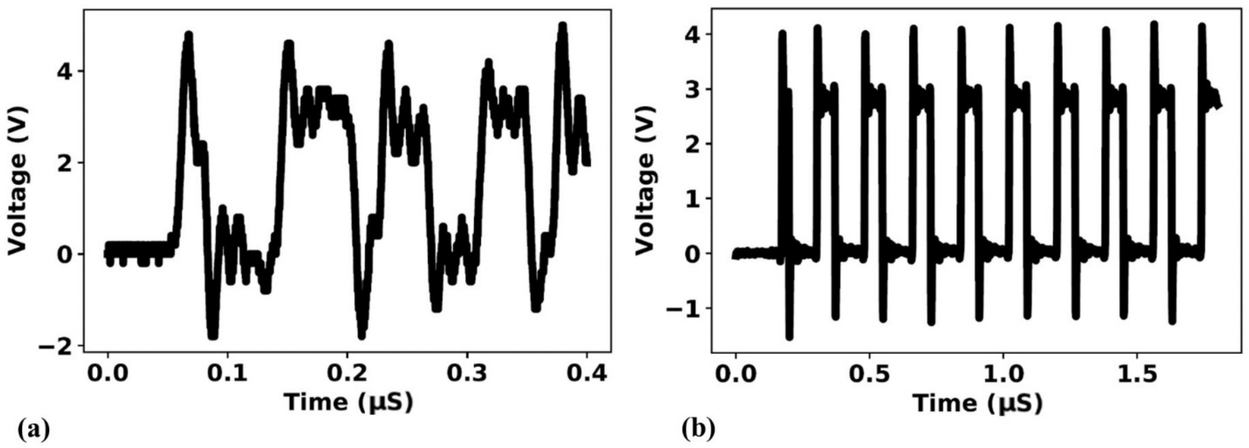


Figure 25. Typical time-domain measurement results of functional leakage-type Trojans: (a) Trojan #3, and (b) Trojan #4.

Table 5. Risk Level Metrics for Trojans on Commercial PCBs

Trigger input	Payload	Signal frequency	Probability of activation (score:0-1)	Degree of payload impact (score: 0-5)
Two USB mouse signals and 3-sec audio play	Computer system freeze	DC	0.002	5
Two USB mouse signals	CPU fan speed change	35 kHz	0.02	2
Two USB mouse signals	Audio bus data leakage	24 MHz	0.02	4
Two USB mouse signals	Ethernet controller data leakage	10 MHz	0.02	4

Next, we replaced the victim HaHa board with commercial boards (initially the motherboard and then the single-board computer). The four Trojans designed for these boards were successfully triggered and resulted in the expected payload impacts (as summarized in Figs. 19 through 22). In the case of Trojan 1, the analog voltage at the temperature sensor input pin of the hardware supervisor chip was converted from 1 V to -1 V, in turn causing the desktop computer to freeze when an audio file was played for more than 3 sec as well as a USB mouse and keyboard were plugged into selected USB ports on the back panel of the motherboard. In the case of Trojan 2, the CPU fan speed was maliciously decreased by ~32% (from 1554 rps to 1051 rps) by triggering a frequency change of the CPU fan connector's PWM input signal (from 50 kHz to 35 kHz) when the two USB devices were plugged in. Trojans 3 and 4 were also inserted into both commercial boards and successfully acquired data bus information from the audio codec and the Ethernet controller, respectively.

The leaked data from the victim boards was successfully probed at the Trojan generator; typical time-domain measurement results are shown in Fig. 25. The signal sensed from Trojan 3 (Fig. 25(a)) is more distorted than that generated by Trojan 4 (Fig. 25(b)) due to its much higher data rate, which makes it susceptible to the effects of cable reflections.

The measurement results were analyzed to derive Trojan risk level metrics, as summarized in Table 5. Potential Trojan threats were assessed using two complementary criteria, namely probability of activation and degree of payload impact, which can be multiplied together to obtain a measure of overall risk. We quantified these criteria using scores with a range of 0 to 1 (for increasing probability of activation) and 0 to 5 (for increasing payload impact). In addition, we also specify the characteristic signal frequency (or data rate) probed by each Trojan. This is because designs that require access to higher-frequency signals are disfavored by attackers due to their increased hardware complexity and data acquisition challenges. Based on our analysis, Trojan 1 has the highest payload impact, because its activation leads to system freezes, which are very undesirable for any computing platform. This Trojan, however, is rarely triggered because its trigger conditions are more stringent than for the others (e.g., based on the number of required USB plug-in events), thus resulting in the lowest probability of activation. By contrast, Trojan 2 has the least serious payload impact, since normal system operation is largely unaffected by a modest change in CPU fan speed. Finally, Trojans 2, 3 and 4 have higher probability of activation because of their more relaxed trigger conditions. However, they are also likely to be disfavored by attackers due to their relatively high signal frequency.

5.7 LIMITATIONS

While the Trojan emulation platform can create benchmarks for assessing and comparing various PCB Trojan countermeasures, there still exist limitations on the types of boards and Trojans (designs, payloads, and triggers) that are supported. One limitation is that leaking sensitive information through power analysis of victim nodes on a commercial PCB is very difficult. The target signal trace path on the PCB has to be broken and a current sensor (e.g., a series resistor or Hall-effect device) added after PCB fabrication and assembly to measure current flow, which is time-consuming and demands significant PCB rework when the board has many layers. Another limitation is that signals from high-speed ICs or peripheral devices on the victim nodes cannot easily serve as Trojan triggers because of two main reasons. Firstly, the FPGA on the HaHa board uses a relatively low clock frequency (in our case, typically set to 50 MHz). Secondly, high-speed I/Os based on low-voltage differential signaling (LVDS) require special impedance-matched driver circuits. In the absence of such drivers, LVDS signals above ~100 MHz are significantly distorted after transmission through 1-2 m of cable from the victim node to either the probing system or the custom HaHa board. The resulting degradation of signal integrity causes loss of the leaked information that in turn makes the Trojan trigger unstable. A final limitation is that Trojans that maliciously write data into on-board memory may not be enabled for some ICs on a commercial motherboard or single-board computer unless the microprocessor or BIOS chip can be reprogrammed. If such specialized knowledge of the firmware is unavailable, inserting Trojans of this type into the PCB may lead to unexpected and permanent system-level failures.

5.8 FLEXIBILITY AND SCALABILITY

The Trojan emulation framework can be extended to find solutions to the three aforementioned problems. Firstly, other options for side channel attacks are available, including acquiring the local magnetic field and/or infrared (thermal) profiles near working ICs and traces. For example, sensing the magnetic fields generated on the PCB with and without Trojans inserted can allow significant amounts of information to be leaked from the victim nodes. In addition, the magnetic field amplitude near a trace is proportional to current flow, which allows power consumption to be calculated. Limitations on information leakage from high-speed signals can be relaxed by replacing the FPGA on the HaHa board (i.e., the Trojan generator) with a higher-end chip that provides LVDS I/O modules for capturing high-speed data. Also, the bandwidth of the high-speed links between the Trojan generator and the victim nodes can be increased by using impedance-matched connections. Finally, collaborative Trojans can be developed to expand the scope of the emulation platform. In this approach, a microcontroller or microprocessor on the commercial PCB can be configured to serve as either the Trojan trigger or payload, thus enabling more complex Trojans to be realized. Fig. 26 shows a possible realization of a complex Trojan using the baseboard management controller (BMC) module, as reported in [8]. The BMC module's function can be implemented into the FPGA of the HaHa board.

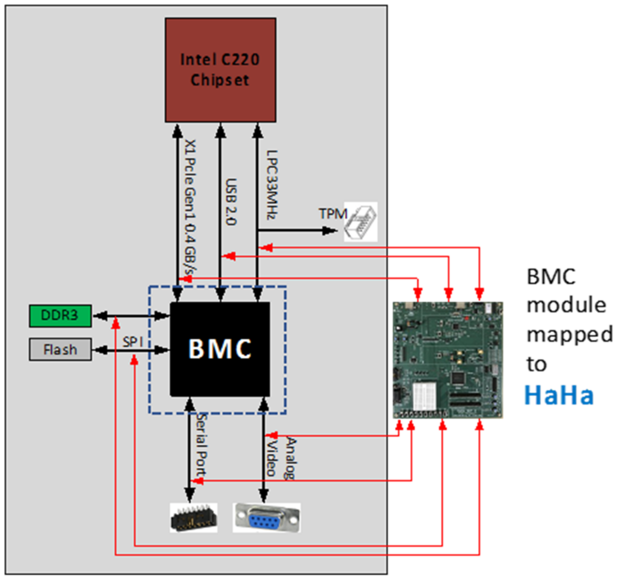


Figure 26. Implementation of the BMC module and other complex trigger logic can be done with the FPGA.

5.9 INTEGRATION WITH EXISTING PCB ANALYSIS TOOLS

More measurement instruments can be integrated into the proposed Trojan emulation platform. For example, a magnetic field probe (e.g., based on Hall-effect sensors) can measure the magnetic field near traces or ICs on the victim PCB, thus enabling power-based side channel attacks as described in the previous section. Also, high-speed digital oscilloscopes or logic analyzers can be added to the setup to allow probing and decoding of high-frequency data buses associated with DDR RAM, graphics processors, and other high-speed components.

6.0 A MACHINE LEARNING BASED GOLDEN-FREE PCB ASSURANCE FRAMEWORK

6.1 TROJAN DETECTION METHODOLOGY

A golden-free Trojan detection framework is necessary for accurately detecting Trojans in a PCB design. We propose a framework that can detect Trojans in a PCB design using Machine Learning (ML) models. The overall flow of the Trojan detection framework is described in Fig. 9 where a Trojan benchmarking tool is used to generate the Trojan inserted designs

For each type of Trojan a set of Trojan inserted designs is generated. The structural and structural-functional features of the Trojan inserted and Benign designs are extracted. The extracted features are represented in the form of an adjacency matrix. The adjacency matrix is flattened to form a feature matrix. The data of the feature matrix is split into train and test dataset comprising of different Benign designs and different Trojan design, having no overlap of Trojan or Design. A machine learning classifier is trained on the training dataset, comprising of Benign and Trojan inserted design features. The trained ML model is evaluated on the test dataset comprising of different Benign and Trojan inserted designs.

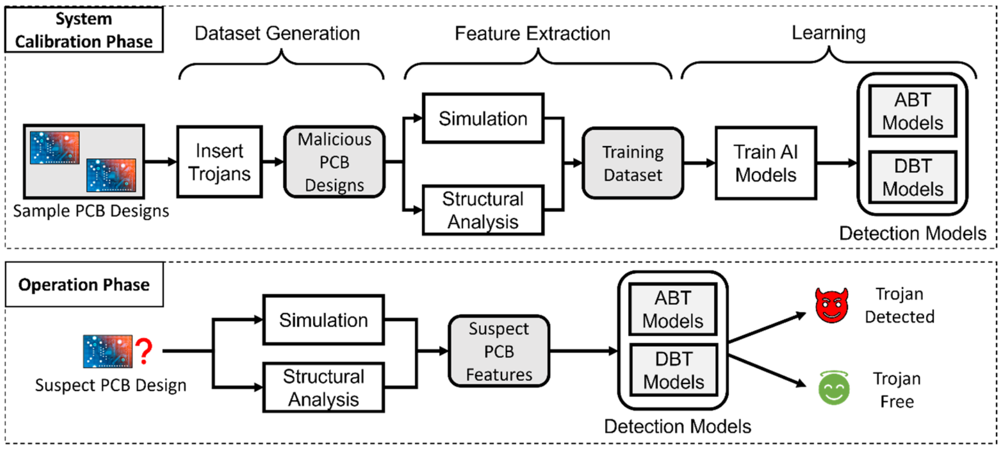


Figure 27. Overview of the proposed framework for hardware Trojan detection using Machine Learning.

6.2 TRAINING DATASET GENERATION

Generating Trojan-inserted PCB designs is one of the primary requirements for an ML-based Trojan detection framework. The Trojan inserted dataset needs to be sufficient as well as the inserted Trojans needs to have variations. We use a dynamic and automated Trojan insertion tool that can insert Trojans in any PCB design. The tool simulates the input design using an open source simulator and the uses it to generate the activation probability of each net in the input design. The structure of the Trojan templates are also modified. Based on the activation probability of the nets in the design, the tool selects the rare nets and connects them to the Trojan Trigger. The Trojan Payload is connected with a randomly selected design net. The Trojan Payload is connected at a random location to increase the coverage of the Trojans across the design.

Using the Trojan insertion tool, a large database of Trojans covering most of the design space can be generated. The Trojan insertion tool also makes the detection method robust by generating structural variations in the Trojan design. The tool can also support new Trojan designs which follow certain input/output rules. This enables us to integrate support for detection of new Trojan designs into the PCB Trojan detection framework.

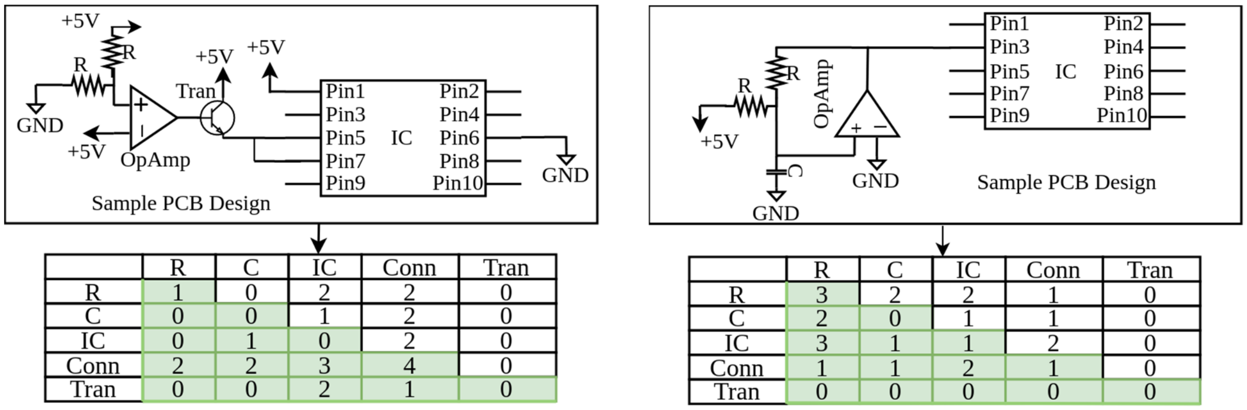
6.3 DATASET PRE-PROCESSING

The Benign and Trojan-inserted PCB designs are pre-processed before extracting the features. Unlike gate level designs, PCB designs do not have a standard library which results in a high variation of component names across PCB designs.

Therefore, we create a macro digest of possible PCB components for labeling and understanding the PCB. The digest consists of 10 component categories: Programmable IC, Non-Programmable IC, Resistor, Capacitor, Crystal, Memory, Inductor, Transistor, and Connector. Components that support a workload or have programmable fabric fall under the Programmable IC category. Components consisting of digital circuits, like Operational Amplifiers, are considered under Non-Programmable IC category. Resistor, Capacitor, and Diode consists of all types of analog resistors, capacitors, and diodes, respectively. Crystals and Oscillators are considered to be Crystals. Filters and Inductors are considered to be Inductor. Memory devices like RAMs are categorized as Memory. Discrete Transistors are considered as Transistors and Input/Output pins are categorized as Connector. While parsing of PCB, every component is annotated with a category from the digest.

6.4 FEATURE EXTRACTION

Based on the pre-processed PCB design description, the framework extracts a set of functional and structural features that are used in the subsequent steps. Fig. 10 illustrates the process of extracting the adjacency matrix from a sample PCB design.



	R	C	IC	Conn	Tran
R	1	0	2	2	0
C	0	0	1	2	0
IC	0	1	0	2	0
Conn	2	2	3	4	0
Tran	0	0	2	1	0

(a)

	R	C	IC	Conn	Tran
R	3	2	2	1	0
C	2	0	1	1	0
IC	3	1	1	2	0
Conn	1	1	2	1	0
Tran	0	0	0	0	0

(b)

Figure 28. The feature extraction process from sample PCB designs: In (a) and (b) the sample PCB design (top) and the corresponding adjacency matrix (bottom) are shown. The part of the adjacency matrix colored in green is flattened to make the feature vector.

6.5 TROJAN DETECTION IN SUSPECTED PCB

The feature matrix generated in the previous section is processed to generate training and testing datasets. The training data is used to train an ML algorithm to classify between Benign and Trojan inserted designs.

During evaluation of ML models among Support Vector Machine (SVM), Logistic Regression and Random Forest (RF), the RF classifier performed the best in detecting Trojans. Hence, we use the Random Forest classifier model for detection of Trojans in a PCB design.

Random Forest (RF) is an ML algorithm based on Decision Tree algorithms. The algorithm utilizes ensemble learning in the prediction of an outcome. Multiple classifiers are used that solve a complex problem. The algorithm is trained using Bootstrap aggregating which generates multiple training sets. Each of the generated Training sets are used to train a particular decision tress. The variance of each of the decision trees might be high but the overall variance is reduced. The reduced variance helps in increasing the stability of the algorithm. The increased stability of the algorithms prevents over-fitting. The average outcome of multiple decision trees are considered for the final prediction.

6.6 EXPERIMENTS AND RESULTS

In this section, we present the evaluation of the proposed PCB-level Trojan detection framework. We procure 10 open-source KiCAD [5] golden designs (see Table 6) and use the automatic Trojan insertion tool proposed in [19] to generate a diverse Trojan inserted design database. For both board level analysis (BLA) and element level analysis (ELA), we first randomly split the golden designs into two bins (train and test) based on a train-test split ratio. Next, we generate 15 variants for each of these design through non-malicious alterations without affecting the functionality. This leads to a total of 150 Trojan-free golden design variants split into test/train bins. For generating the Trojan inserted designs, we insert 2 to 6 ABT/DBT Trojans in the original 10 golden designs and split it based on the same train-test split ratio. For experimenting with partial PCB designs we first use hmetis to generate a random number of partitions in the range of [8,20]. Subsequently, a set of [1,8] partitions are considered hidden during the analysis.

6.6.1 BOARD LEVEL ANALYSIS (BLA)

In Table 7, for DBT with complete designs, we observe >90% Trojan/non-Trojan classification accuracy, Trojan class precision, and Trojan class recall. The same values are more than 80%, for ABT with complete designs. We observe a slight drop in detection efficiency for partial designs. Also, structural (struct) and functional (func) features combined generally leads to better accuracy for both ABT and DBT. Note, that the recall of our models are generally high indicating low false negatives. ASIC-based Trojans with only structural features appear to be harder to detect because they generally have a higher similarity with the host design. However, the detection efficiency increases when both structural and function features are combined.

6.6.2 ELEMENT LEVEL ANALYSIS (ELA)

For element level analysis (see Table 8), the accuracy, precision and recall values are generally high (>90%) expect for ABT with only structural features. As explained earlier, ABTs have a feature space overlap with the host design hence detecting them is slightly harder. As this is an element level analysis, if the framework can detect most of the malicious PCB components/elements, then the undetected/mislabeled malicious components can be subsequently inferred through manual inspection.

6.6.3 TRAINING DATA REQUIREMENT ANALYSIS

Obtaining good quality training data for PCB Trojan detection can be challenging. Hence, in Fig. 27 and Fig. 28 we observe the effect of training dataset size on the detection accuracy. Increasing the training dataset leads to higher detection efficacy, however, even with a 60% training dataset, we were able to obtain >75% accuracy for all scenarios with only structural features. We observe similar trend for with both structural and functional features.

Table 6. Sample PCB Designs for Inserting Trojans

Name of Design	No. of comp.	No. of comp.	No. of conn.	DC based Trojans inserted	ASIC based Trojans inserted
CIAA ACC	569	12	787	15	10
EDU CIAA K60	187	2	238	15	10
CIAA K60	467	4	407	15	10
CIAA FSL-MINI	141	4	188	15	10
CIAA-PIC	433	4	404	15	10
EDU-INTEL	87	2	150	15	10
CIAA PICO	67	2	144	15	10
CIAA SAFETY	275	2	240	15	10
CIAA-Z3R0	33	2	52	15	10
Audio Amp	49	2	78	15	10

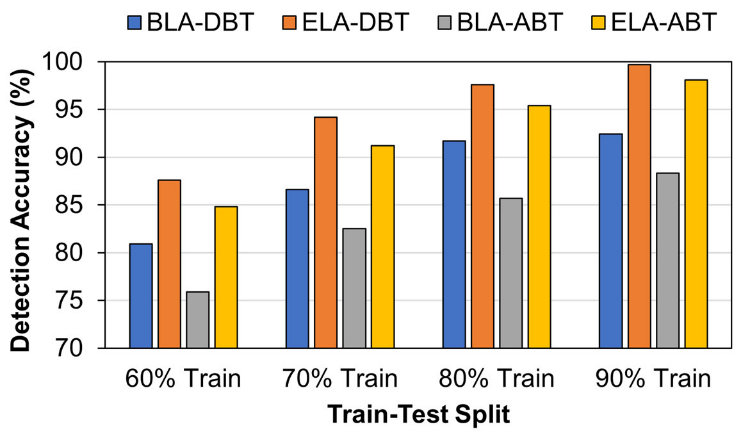


Figure 29. Overall detection accuracy across all scenarios for different training dataset sizes with structural features.

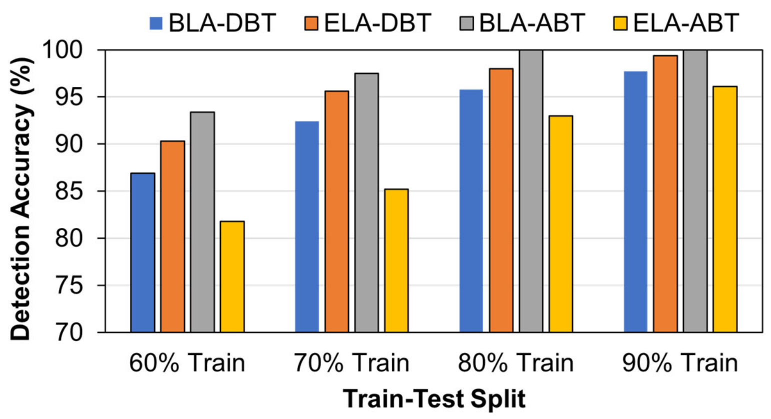


Figure 30. Overall detection accuracy across all scenarios for different training dataset sizes with both structural and functional features.

Table 7. Trojan detection results for board level analysis

Trojan Type	Features Used	Accuracy (%)		Trojan Class Precision (%)		Trojan Class Recall (%)	
		Com-plete Design	Partial Design	Com-plete Design	Partial Design	Com-plete Design	Partial Design
DBT	Struct	91.7	89.2	93.6	91.3	94.8	93.5
	Struct + Func	95.8	92.1	94.5	94.6	99.7	99.3
ABT	Struct	85.7	68.6	85.4	71.0	100	97.0
	Struct + Func	86.2	84.3	86.2	82.9	100	90..4

Table 8. Trojan detection results for element level analysis

Trojan Type	Features Used	Accuracy (%)		Trojan Class Precision (%)		Trojan Class Recall (%)	
		Com-plete Design	Partial Design	Com-plete Design	Partial Design	Com-plete Design	Partial Design
DBT	Struct	97.6	94.8	94.9	92.5	96.2	92.6
	Struct + Func	98.0	95.6	95.0	93.2	95.7	93.7
ABT	Struct	85.4	89.6	76.3	90.7	87.6	86.5
	Struct + Func	86.7	92.8	95.1	92.4	91.4	88..4

7.0 CONCLUSIONS

In this seedling project, we have developed a PCB-level Trojan benchmarking system and explored the feasibility of a machine learning based golden-model-free PCB trust verification process. In this final report, we have described the major findings related to PCB-level Trojan insertion and PCB trust verification. We have presented a taxonomy for PCB-level malicious circuits with associated designs of hardware Trojans. We have also presented a software tool flow for automatically generating a large number of diversified Trojan-inserted PCB benchmarks with different configuration options for Trojan insertion. We have analyzed the feasibility of these designs by fabricating a set of representative Trojans on a custom PCB and observing their side-channel footprint. We also analyzed the stealthiness of the Trojans through structural analysis and accessed the runtime by generating hundreds of PCB Trojan benchmarks.

Moreover, we have described the design of a flexible hardware emulation platform that enables Trojan insertion and validation on both custom-made and commercial PCBs. A set of specialized hardware Trojans for both boards have also been designed and implemented. We have designed the Trojan triggers and payload circuits for activation using both internal (i.e., onboard) and external signals in both cases. We have implemented and presented an FPGA-based Trojan generator using the custom HaHa board to achieve this. The resulting data were analyzed to derive benchmarks for PCB-level Trojans, thus exploring their effects without fabricating and testing many board variants.
Through hands-on experiments, we have exhibited that the Trojans triggered on HaHa boards can invert the state of an LED or leak the LED's state information to the Trojan generator. In addition, the Trojan-inserted motherboards and single-board computers can be triggered automatically through multiple mechanisms (e.g., via USB device plug-in events or audio signals) to achieve the desired payload effects (e.g., system freezing, malicious changes in CPU fan speed, or sensitive data leakage from audio codecs or Ethernet controllers). The experimental results were analyzed to derive a risk metric for Trojans on commercial boards. The analysis manifested that Trojans that generate the most severe outcomes (e.g., system freezing) are less prone to be triggered (rare), thus reducing their risk of being exposed. On the contrary, the information leakage Trojans can be activated more often; however, they have less impact on the system.

Finally, we have presented a comprehensive ML-based PCB-level Trojan detection framework that eliminates the need for golden or reference models. It integrates an automatic Trojan insertion tool for creating a training dataset capable of creating significant structural and functional variants of Trojan instances for various Trojan classes. It uses a combination of structural (represented as subgraphs) and functional (computed as activation probabilities) features for training the ML models. The proposed method can detect whether a PCB is malicious and determine the type and location of malicious components present on a PCB. We have developed a completely automated tool flow, interfaced with commercial PCB design flow (with Ngspice for functional simulation), and evaluated it extensively with diverse PCB designs and Trojan types. We observed that the proposed approach is highly effective in detecting unknown design alterations, and the false positive rate is very insignificant (*<5%*). The method is scalable to large PCB designs and flexible to account for emerging Trojan types by accounting for a new Trojan template during training dataset creation.

While this seedling effort has made significant progress on PCB assurance under hardware Trojan attacks and demonstrated the promise of a golden-free trust verification framework, our future research and next steps can lead to transition of this technology to ensure trustworthiness of PCBs used in diverse DoD electronic systems, considering an untrusted supply chain. Future research on this topic will include maturation of the PCB Trojan insertion tool and hardware emulator as well as development of a golden-model-free PCB assurance framework that effectively leverages the power of supervised machine learning models. It will include development of always-on Trojans for PCBs with low area footprint and automatic insertion of Trojans from alteration model that does not require incorporating new components. Automatic insertion of Trojan components between layers at the layout level using the Oreo construction can also be explored. Countermeasures that use reverse-engineering and side-channel analysis of power and thermal profiles can be evaluated using encapsulated Trojans with a small area footprint. We will also focus on improving the Trojan detection capability by considering additional features and more powerful ML models and evaluating emerging Trojan types.

8.0 REFERENCES

[1] S. Ghosh, A. Basak and S. Bhunia, "How secure are printed circuit boards against Trojan attacks?," *IEEE Design & Test,* vol. 32, p. 7–16, 2014.

[2] B. Shakya, T. He, H. Salmani, D. Forte, S. Bhunia and M. Tehranipoor, "Benchmarking of hardware trojans and maliciously affected circuits," *Jour. of Hardware and Syst. Security,* vol. 1, p. 85–102, 2017.

[3] J. Cruz, Y. Huang, P. Mishra and S. Bhunia, "An automated configurable Trojan insertion framework for dynamic trust benchmarks," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2018.

[4] H. Zhu, X. Guo, Y. Jin and X. Zhang, "PCBench: Benchmarking of Board-Level Hardware Attacks and Trojans," in *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2021.

[5] *KiCad EDA: A Cross Platform and Open Source Electronics Design Automation Suite,* 2021.

[6] S. Paley, T. Hoque and S. Bhunia, "Active protection against PCB physical tampering," in *2016 17th International Symposium on Quality Electronic Design (ISQED)*, 2016.

[7] *OREO CONSTRUCTION: HIDING YOUR COMPONENTS INSIDE THE PCB,* 2019.

[8] J. Robertson and M. Riley, "The Big Hack: How China Used a Tiny Chip to Infiltrate U.S. Companies," *Bloomberg Businessweek,* 2018.

[9] *Modchips of the State,* 2020.

[10] M. Steil, "17 mistakes Microsoft made in the Xbox security system," in *22nd Chaos Communication Congress*, 2005.

[11] *NSA ANT PRODUCT CATALOG,* 2013.

[12] K. Rosenfeld and R. Karri, "Attacks and Defenses for JTAG," *IEEE Design & Test of Computers,* vol. 27, p. 36–47, 2010.

[13] M. McGuire, U. Ogras and S. Ozev, "PCB Hardware Trojans: Attack modes and detection strategies," in *2019 IEEE 37th VLSI Test Symposium (VTS)*, 2019.

[14] K. Yang, M. Hicks, Q. Dong, T. Austin and D. Sylvester, "A2: Analog malicious hardware," in *2016 IEEE symposium on security and privacy (SP)*, 2016.

[15] *An Easier and Powerful Online PCB Design Tool,* 2021.

[16] *CIAA: Computadora Industrial Abierta Argentina,* 2020.

[17] *CIAA-PicoCIAA: The CIAA Project,* 2018.

[18] M. Berlingerio, D. Koutra, T. Eliassi-Rad and C. Faloutsos, "Netsimile: A scalable approach to size-independent network similarity," *arXiv preprint arXiv:1209.2684,* 2012.

[19] T. Hoque, S. Yang, A. Bhattacharyay, J. Cruz and S. Bhunia, "An Automated Framework for Board-level Trojan Benchmarking," *ArXiv,* vol. abs/2003.12632, 2020.

[20] M. C. S. B. Lipner, "A Comment on the Confinement Problem," 1975.

[21] J. McLean, "Security Models," in *In Encyclopedia of Software Engineering 2*, New York, John Wiley & Sons, 1994, p. 1136–1145.

[22] B. Fletcher, "DISA," 17 12 2008. [Online]. Available: http://iase.disa.mil/cds/helpful_tools/dfcf_specification_1_2_10.pdf. [Accessed 17 12 2008].

[23] M. Atighetchi, "XDDS Installation, Administration, and Demonstration Guide," Cambridge, 2009.

[24] O. U. S. TC, "Using WSDL in a UDDI Registry - Version 2.0.2 - Technical Note," 31 June 2004. [Online]. Available: http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-wsdl-v2.htm.

[25] A. V. a. P. Bonderud, "jUDDI," 24 12 2008. [Online]. Available: http://ws.apache.org/juddi/. [Accessed 24 12 2008].

[26] ISO, "ISO/IEC 19757 - Document Schema Definition Languages (DSDL) - Part 3: Rule-based validation - Schematron," 2006.

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

DOD	Department of Defense