5-2021

Characteristic Reassignment for Hardware Trojan Detection

Noah Waller
University of Arkansas, Fayetteville

Characteristic Reassignment for Hardware Trojan Detection

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Engineering

by

Noah Waller
University of Arkansas
Bachelor of Science in Computer Engineering, 2019

May 2021
University of Arkansas

This thesis is approved for recommendation to the Graduate Council.

____
Jia Di, Ph.D.
Thesis Director

Qinghua Li, Ph.D.	Dale Thompson, Ph.D.
Committee Member	Committee Member

ABSTRACT

With the current business model and increasing complexity of hardware designs, third-party Intellectual Properties (IPs) are prevalently incorporated into first-party designs. However, the use of third-party IPs increases security concerns related to hardware Trojans inserted by attackers. A core threat posed by Hardware Trojans is the difficulty in detecting such malicious insertions/alternations in order to prevent the damage. This thesis work provides major improvements on a soft IP analysis methodology and tool known as the Structural Checking tool, which analyzes Register-Transfer Level (RTL) soft IPs for determining their functionalities and screening for hardware Trojans. This is done by breaking down primary ports and internal signals into assigned assets that are spread out into six characteristics. Using characteristics based on the external primary ports and the internal signals connected to them, reassignment of assets can be used to match against entries using coarse-grained-to-coarse-grained matching against a subset of known-IPs to classify an unknown soft IP. After determining the soft IP's functionality, asset reassignment occurs within the Golden Reference Library (GRL), a library consisting of known Trojan-free and Trojan-infested entries. A fine-grained-to-fine-grained asset reassignment is used against the GRL to contain the most up-to-date assets based on the unknown soft IP, where the matching process is used to determine if the soft IP is Trojan-free or Trojan-infested. With the increasing size of the GRL, the need to decrease computational resources while also maintaining high accuracy between unknown soft IPs and GRL entries is vital.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Jia Di. He has encouraged me to become a better leader and supported me throughout my time as an undergraduate and graduate student. He is a great mentor and I thank him for his guidance.

I would also like to thank the members of my committee members Dr. Dale Thompson and Dr. Qinghua Li for their guidance throughout my undergraduate and graduate careers.

I would finally like to thank Hunter Nauman, Rafael Del Carmen, Derek Taylor, and Flora Smith for their help and support during my graduate career.

DEDICATION

To my Mom, Dad, and brothers for supporting me throughout my life. They have encouraged me to pursue my dreams and have always given me the support I needed to achieve this degree.

CONTENTS

1. INTRODUCTION

Due to economic considerations, the number of third-party hardware Intellectual Property (IP) vendors has increased significantly. It is not financially feasible to design every component of an Integrated Circuit (IC) IP in-house, so first-party IC vendors contract third-party vendors to design certain components. By doing so, the integrity of the overall soft IP can be compromised. The insertion of a hardware Trojan into a third-party component can be fairly easy, posing threats to critical applications. A hardware Trojan is defined as "a malicious, intentional modification of a circuit design that results in undesired behavior when the circuit is deployed" [1]. These payloads include leaking of cryptographic keys from encryption units, tampering of data, and denial of service for devices.

Numerous solutions have been proposed that focus on hardware Trojan detection. One area of research is side-channel analysis. This type of analysis focuses on the natural emissions of a circuit, including power and timing delays, to reveal any possible modifications. In [2], internal impedance reflected from an integrated circuit was used to nondestructively detect for hardware Trojans. One issue with using a reflected internal impedance is the inserted Trojan has a possibility of being sufficiently small such that it is unnoticeable when comparing against the impedance of the entire circuit. In [3], a technique from differential power analysis was used to detect hardware Trojans. This technique differs from [2] in that it uses power analysis, which can be more sensitive to small-footprint circuits, rather than impedance for detection of hardware Trojans. However, similar to using impedance, the hardware Trojan may still not be large enough for detection, and the manufacturing process between two chips may lead to false positives. While these two techniques are valid methods of detecting Trojans, they are both limited to detecting Trojans on physical chips, known as hard IPs.

A second area of research for detection of hardware Trojans is at the soft IP level. Soft IPs are Register-Transistor-Level (RTL) code or other gate-level netlists. One strategy for detecting hardware Trojans using gate-level netlists was developed in [4] where the use of natural language processing and statistical analysis distinguishes the “naturalness” of a circuit against the “unnaturalness” of a hardware Trojan. A second strategy using gate-level netlists was introduced in [5]. Machine learning is used on net testability and netlist structural features found within gate-level netlists to detect the instance of a possible hardware Trojan.

Different from the research for soft IPs mentioned above, the Golden Reference Matching method in [6] uses RTL code, rather than gate-level netlists. Golden Reference Matching breaks apart RTL code into components and primary ports. Then, internal signals are labeled using assets which describe the functionality of the overall soft IP. Once assets have been assigned to the unknown soft IP, it is compared against a Golden Reference Library (GRL). This GRL contains a collection of entries that are known to be either Trojan-free or Trojan-infested. Once compared against the entries within the GRL, the soft IP with the highest match to an entry is categorized. If the unknown soft IP best matches against a Trojan-infested entry, then the IP likely contains a Trojan. Conversely, if the unknown soft IP best matches against a Trojan-free entry, then the IP likely does not contain a Trojan.

To decrease computational resource usage while preserving categorization of soft IPs as Trojan-infested or Trojan-free, a subset of entries within the GRL is taken and used as a champion entry to be used in a newly developed Champion GRL. The champion entry used is considered the best entry of a functionality and is used initially in matching, where an unknown soft IP is given the functionality of the highest match within the Champion GRL. With the limited number of designs within the Champion GRL, external assets are generalized into 10 categories and are reassigned to the unknown soft IP to increase matching between functionalities. After a functionality is given to the IP, it is matched to designs only within its functionality using the Functionality GRL, a GRL broken apart into different functionalities. This method of matching decreases computational usage, allowing for more unknown IPs to be categorized while preserving a high confidence in categorization of functionalities as well as if the IP is Trojan-infested or not.

The rest of the thesis is organized as follows. Chapter 2 will cover background information on assets, Structural Checking, and Golden Reference Matching with a Golden Reference Library. Chapter 3 will cover the design and implementation of asset reassignment through coarse-grain and fine-grain applications. Chapter 4 will provide examples of soft IPs to prove the effectiveness of the improved matching process. Chapter 5 will conclude the thesis and provide details on future work.

2. BACKGROUND

2.1 Assets

2.1.1 Overview

Assets are critical to the Structural Checking tool with respect to Golden Reference Matching, which is explained more in-depth later in this thesis. These assets provide labels to both the primary ports as well as other internal signals of soft IPs about their purpose/function in the context of the design hierarchy. Each signal can have multiple asset labels assigned to it to improve the description of the overall design. The Structural Checking tool utilizes two main categories of assets, internal assets, and external assets.

2.1.2 Internal Assets

Internal assets are assets describing the function of internal signals in a soft IP but can describe primary port signals as well. Most internal assets used in the tool were developed in [7] and [8]. The internal assets developed in [7] are automatically assigned as the Structural Checking tool parses through Register-Transfer Level (RTL) code. Other internal assets, specifically those developed in [8], differ, as they are manually assigned by the user.

2.1.3 External Assets

External Assets are assets identifying the purpose of primary ports in a soft IP. These are manually assigned to each primary port signal with the Structural Checking tool. The majority of external assets were developed in [6] and [7] and were grouped into 5 categories: *Data, Timing, System Control, Specific System Control,* and *Miscellaneous*. Each category encompasses signals contributing to the domain of the given category. Assets falling in the *Data* category pertain to the flow of data through a circuit, whereas assets located inside of the *Timing* category pertain to the timing of a circuit. *System Control* and *Specific System Control* assets relate to the control of a circuit. The *System Control* category includes more general assets such as *RESET*, *READ*, and *ENABLE*, while the *Specific System Control* category includes niche control assets used in specific circuits such as *DATA_OP*, *COMMUNICATION_CONTROL*, and *INTERRUPT_CONTROL*. Finally, *Miscellaneous* assets refer to any other types of assets that may be defined within a circuit.

2.1.4 Asset Filtering

The goal of asset filtering was introduced in [9] and is used to propagate the assigned assets of a signal through signals connected to it. Propagating these assets allows the tool to find correlations between signals as well as conflicting assets assignments. External assets serving as primary circuit inputs propagate to any related primary outputs. External assets assigned to primary circuit outputs propagate backwards to associated primary inputs. However, internal assets contain a few exceptions to this method of asset filtering. For example, when filtering a process-sensitive internal asset, the asset propagation only affects signals connected to the original signal and contained within the same process block. Conditional assets also do not adhere to the propagation norm. These assets, like process sensitive assets, only propagate within their conditional statements. The remaining internal assets work within concurrent statements and follow the same asset filtering process as external assets.

2.1.5 Asset Pattern and Characteristics

The set of all assets assigned to a given signal is called an asset trace, introduced in [6]. Asset traces of a soft IP are then collected into an asset pattern. Asset patterns include 6 characteristics. External assets assigned to primary input port signals form a single characteristic, denoted by “>”, while internal assets assigned to an external signal form a second characteristic, denoted by “>*”, both located within a GRL file. External assets assigned to primary output ports are denoted by “<”, and internal assets assigned to primary output ports are denoted by “<*”. External assets assigned to internal signals are denoted by "/", and internal assets assigned to internal signals are denoted by "/*". After the asset pattern is created, it is stored into an asset file.

2.2 Golden Reference Matching

2.2.1 Overview

Developed in [6], Golden Reference Matching processes an unknown soft IP by comparing it against a Golden Reference Library (GRL) containing known Trojan-free and Trojan-infested soft IPs, and it determines whether the unknown soft IP contains Trojans. For every entry in this Golden Reference Library, the algorithm behind the matching process calculates a percent match against the unknown soft IP by comparing asset similarity between the two. Based upon the highest percent match of the unknown soft IP against the Golden Reference Library entries, Golden Reference Matching provides a probabilistic result indicating both the general functionality of the soft IP and whether the unknown soft IP may contain Trojans.

2.2.2 Basic Matching

Basic matching consists of a percentage match of the asset characteristics based on the asset trace of the unknown soft IP and the GRL entries. Comparing all assets in an asset trace will provide a percent match for the trace of the given characteristic. The percent matches of all traces are then averaged and determine an overall percent match for the specific characteristic. Once all characteristics are matched, the average of the 6 percent matches is leveraged to calculate an overall match for the unknown soft IP. However, there are special cases where either the unknown soft IP or the GRL entries do not comprise assets in each characteristic. In this instance, the empty characteristic is excluded from the overall percent match calculation.

Table 1: Basic Matching Example

Trace	Unknown IP Assets	GRL Entry Assets	Percent Match
1	DATA_COMPUTATIONAL	DATA_COMPUTATIONAL	100%
2	DATA_SENSITIVE,	DATA_MEMORY	0%
3	DATA_SENSITIVE, READ, WRITE, LOAD	DATA_SENSITIVE, HOLD, COUNT, SHIFT	25%

Table 1 provides example outcomes of the matching process. Each row of Table 1 contains asset traces of a single characteristic from an unknown IP and asset traces of the same characteristic from a GRL entry. Trace 1 demonstrates a 100% match since both assets in this trace are identical. Trace 2 results in a 0% match since the assets share no commonalities. Trace 3 produces a 25% match, as it can only match 1 of 4 assets between the soft IP and corresponding GRL entry. The percent match for the total characteristic is 41.66%.

2.2.3 Partial Matching

Partial matching was introduced to the Structural Checking tool in [6]. When assets are not identical but share a similar purpose in a soft IP, a 50% match is assigned to the two assets. Furthermore, the Basic Matching algorithm was altered to provide a partial match if an asset from either the unknown soft IP or the GRL entry was generic while the other was specific.

Table 2: Partial Matching Example

Trace	Unknown IP Assets	GRL Entry Assets	Percent Match
1	DATA_COMPUTATIONAL	DATA_COMPUTATIONAL	100%
2	DATA_SENSITIVE,	DATA_MEMORY	50%
3	DATA_SENSITIVE, READ, WRITE, LOAD	DATA_SENSITIVE, HOLD, COUNT, SHIFT	25%

Table 2 provides the same examples from Table 1. Using partial matching, however, trace 2 receives a 50% match because *DATA_SENSITIVE* is the generic version of the *DATA_MEMORY* asset. Thus, the overall percent match for this characteristic is 58.33%.

2.2.4 Asset Reassignment

Reassignment of a specific asset label to a more generalized asset label is utilized as introduced in [10]. This idea stems from Section 2.2.3, where when a more specific asset is matched with its generic counterpart, as two signals could theoretically be the same, but due to certain assets not having been introduced in earlier stages, a generic asset was assigned to the given signal. If the two assets are found within the same category while comparing a general asset and a specific asset, the specific asset is reassigned to as a general asset for a higher comparison, meaning the specific asset is reassigned to the general asset and a percent match for the assets is 100%. For example, the *DATA_COMPUTATIONAL* and *DATA_SENSITIVE* assets are both located within the *Data* asset category. The *DATA_COMPUTATIONAL* asset would be reassigned as *DATA_SENSITIVE*, and the percent match between these two assets would become 100% as opposed to 50%.

Table 3: Asset Reassignment Example

Trace	Unknown IP Assets	GRL Entry Assets	Percent Match
1	DATA_COMPUTATIONAL	DATA_COMPUTATIONAL	100%
2	DATA_SENSITIVE,	DATA_MEMORY	100%
3	DATA_SENSITIVE, READ, WRITE, LOAD	DATA_SENSITIVE, HOLD, COUNT, SHIFT	25%

Table 3 again depicts the same examples from Table 1 and Table 2. Using asset reassignment, trace 2 has a 100% match because like table 2, *DATA_SENSITIVE* is a generic version of the *DATA_MEMORY* asset. Therefore, the new overall percent match for this characteristic is 75%.

2.2.5 Statistical Matching

In [10], statistical matching was added to the Structural Checking Tool's matching algorithm. Assets that are included a single characteristic of numerous GRL entries should have a lower matching weight compared to assets that are only found within a small subset. An average asset weight is calculated based on the sum of the matched asset weights divided by the total number of matched assets within the characteristic.

$$Weight_{char} = \frac{Characteristic_{char}.AverageAssetWeight}{\sum_{i=A}^{F} Characteristic_i.AverageAssetWeight} * 100$$

Figure 1: Characteristic Weight Calculation [10]

Figure 1 demonstrates the calculation for the total weight of a characteristic. Once the average asset weight is determined for the characteristic, it is divided by the sum of all characteristics' average asset weights. The quotient is then converted to a percentage based on the sum of the 6 characteristics' average asset weight within the Golden Reference Library.

2.2.6 Golden Reference Library

The GRL is a collection of soft IPs retrieved from Trust-Hub [11, 12] and OpenCores [13]. All entries located in the GRL are processed by the Structural Checking tool. An asset pattern is then generated from the tool, and a general functionality is associated with the file to label the overall function of the soft IP. Table 4 lists the functionalities encompassed in the GRL.

Table 4: Functionalities

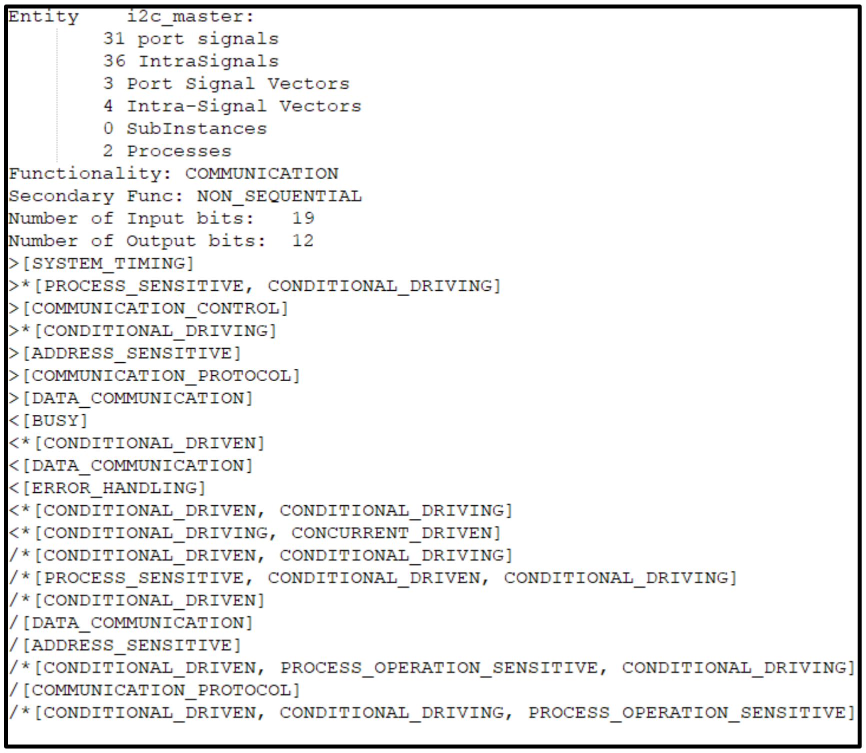
Whitelist Functionality	**Blacklist Functionality**
SHIFT_REGISTER	TROJAN_ENCYPTION_UNIT
INTERRUPT_UNIT	TROJAN_TRIGGER
COMMUNICATION	TROJAN_COMMUNICATION
ENCRYPTION_UNIT	TROJAN_SHIFT_REGISTER
COMPUTATIONAL	
TIMING	
CONTROL_GENERATION	
REGISTER_FILE	
PERIPHERAL	
DECODER_ENCODER	
DEBUG_INTERFACE	

Table 4 (Cont.)

Whitelist Functionality	Blacklist Functionality
TOP_CONTROLLER	
TOP_PROCESSOR	

All GRL entries in the tool are appropriately labeled in terms of being Trojan-Free (Whitelist) or Trojan-infested (Blacklist), as all designs are well-documented and come from trusted sources. If an unknown soft IP matches best with a Whitelisted functionality, it is labeled both as that functionality and as Trojan-free. If an unknown soft IP matches best with a Blacklisted functionality, it is labeled with the same functionality and is subsequently flagged as potentially containing a Trojan.

Figure 2 provides an example of a GRL entry. The top of the file contains the entity's name along with the number of signals the entity contains, any sub-instances, and any processes located within the entity. Afterwards, the file is assigned a functionality, which is *Communication* The remainder of the file contains the asset pattern of the entry to be employed during the matching process.



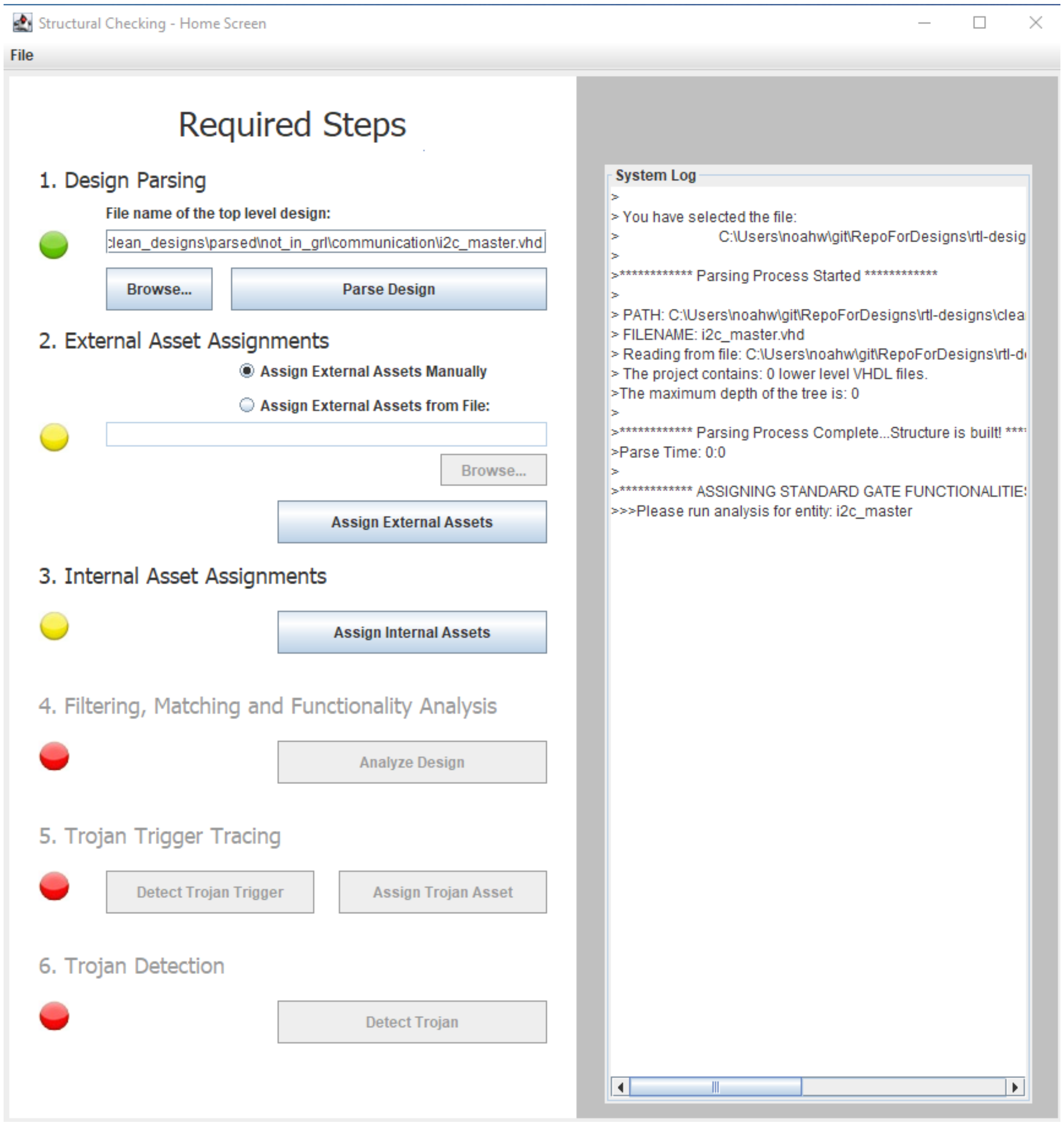
```
Entity    i2c_master:
        31 port signals
        36 IntraSignals
        3 Port Signal Vectors
        4 Intra-Signal Vectors
        0 SubInstances
        2 Processes
Functionality: COMMUNICATION
Secondary Func: NON_SEQUENTIAL
Number of Input bits:   19
Number of Output bits:  12
>[SYSTEM_TIMING]
>*[PROCESS_SENSITIVE, CONDITIONAL_DRIVING]
>[COMMUNICATION_CONTROL]
>*[CONDITIONAL_DRIVING]
>[ADDRESS_SENSITIVE]
>[COMMUNICATION_PROTOCOL]
>[DATA_COMMUNICATION]
<[BUSY]
<*[CONDITIONAL_DRIVEN]
<[DATA_COMMUNICATION]
<[ERROR_HANDLING]
<*[CONDITIONAL_DRIVEN, CONDITIONAL_DRIVING]
<*[CONDITIONAL_DRIVING, CONCURRENT_DRIVEN]
/*[CONDITIONAL_DRIVEN, CONDITIONAL_DRIVING]
/*[PROCESS_SENSITIVE, CONDITIONAL_DRIVEN, CONDITIONAL_DRIVING]
/*[CONDITIONAL_DRIVEN]
/[DATA_COMMUNICATION]
/[ADDRESS_SENSITIVE]
/*[CONDITIONAL_DRIVEN, PROCESS_OPERATION_SENSITIVE, CONDITIONAL_DRIVING]
/[COMMUNICATION_PROTOCOL]
/*[CONDITIONAL_DRIVEN, CONDITIONAL_DRIVING, PROCESS_OPERATION_SENSITIVE]
```

Figure 2: I^{2}C Master GRL Entry

2.3 Structural Checking GUI

The Structural Checking methodology was implemented using a Java-Based GUI. This GUI allows a user to navigate to a VHSIC Hardware Description Language (VHDL) file to be parsed, have assets assigned to signals, to filter assigned assets throughout the circuit, and to be matched with entries within the GRL. The GUI is shown below in Figure 3.

The left side of Figure 3 shows six steps: design parsing, external asset assignment, internal asset assignment, filtering – matching – functionality analysis, Trojan trigger tracing, and Trojan detection. The dot on the left side of the screen is an indication of each step. The red dot indicates that the previous step is incomplete. The yellow dot indicates the step the user is on. The green dot indicates that the step is complete. The right side of the is the system log screen to display information to the user.


Structural Checking - Home Screen
File
Required Steps
1. Design Parsing
File name of the top level design:
Browse...
Parse Design
2. External Asset Assignments
Assign External Assets Manually
Assign External Assets from File:
Browse...
Assign External Assets
3. Internal Asset Assignments
Assign Internal Assets
4. Filtering, Matching and Functionality Analysis
Analyze Design
5. Trojan Trigger Tracing
Detect Trojan Trigger
Assign Trojan Asset
6. Trojan Detection
Detect Trojan
System Log

File name of the top level design: :lean_designs\parsed\not_in_grl\communication\i2c_master.vhd

(●) Assign External Assets Manually
( ) Assign External Assets from File:

```
>
> You have selected the file:
>                 C:\Users\noahw\git\RepoForDesigns\rtl-desig
>
>************ Parsing Process Started ************
>
> PATH: C:\Users\noahw\git\RepoForDesigns\rtl-designs\clea
> FILENAME: i2c_master.vhd
> Reading from file: C:\Users\noahw\git\RepoForDesigns\rtl-d
> The project contains: 0 lower level VHDL files.
>The maximum depth of the tree is: 0
>
>************ Parsing Process Complete...Structure is built! ***
>Parse Time: 0:0
>
>************ ASSIGNING STANDARD GATE FUNCTIONALITIE
>>>Please run analysis for entity: i2c_master
```

Figure 3: Structural Checking main GUI

3. METHODOLOGY AND IMPLEMENTATION

3.1 Overview

The Structural Checking matching process described in Chapter 2 of this thesis leads to an inefficient use of computational resources. As the tool has improved with the Golden Reference Library gaining over 160 trusted entries for unverified IP comparison, the matching process itself has not been altered. As a result, when a new, unknown soft IP is introduced to the tool, the soft IP is compared against all GRL entries, leading to an increase in computational and memory resources. To address these shortcomings, a so-called Champion GRL consisting of a single entry from every whitelisted functionality within the GRL was incepted. When an unknown soft IP is matched using this Champion GRL, a whitelisted functionality is assigned to the soft IP based on the highest percentage match. Once a matching functionality is determined, the original GRL is partitioned into distinct functionalities, so the soft IP can match against entries of the same functionality. Pairing two fine-grained assets is the most precise way to determine if an unknown soft IP is Trojan-infested.

3.2 Champion Golden Reference Library Matching

3.2.1 Champion Golden Reference Library

To establish general functionality for the unknown soft IP while conserving computational resources, GRL entries are inspected manually, and a subset of entries are copied and included in a separate library. Entries containing too few asset traces as well as too few specific assets yield a bias caused by asset reassignment of specific assets to general assets. For instance, multiple entries in the Champion GRL contain the *DATA_SENSITIVE* asset while the unknown soft IP contains the *DATA_COMPUTATIONAL* asset. When performing asset reassignment, the *DATA_COMPUTATIONAL* asset becomes *DATA_SENSITIVE*, and all entries including that asset will have an imprecise match of 100%. However, designs featuring many asset traces with multiple specific assets may exhibit bias when the unknown soft IP contains fewer specific assets than a matching Champion entry, also adversely impacting matching accuracy. For example, a single asset trace of a Champion entry's characteristic consists of *DATA_COMPUTATIONAL*, *PROGRAM_COUNTER_OP*, *DATA_OP*, *SHIFT*, and *STATE* assets while the unknown soft IP only contains the *DATA_COMPUTATIONAL* asset, resulting in a 20% match for the given characteristic. Entries were gathered and their asset traces are analyzed, and a single entry from every whitelisted functionality that best represents it is added to the Champion GRL.

3.2.2 Coarse-Grained Asset Reassignment

Because of entry-limitations of the Champion GRL, a coarse-grained-to-coarse-grained asset reassignment set was also introduced. Due to the fine-grained comparisons of the unknown IP and the Champion GRL entries, top Champion GRL matches have a lower matching percentage with the soft IP when compared against a single design within the same functionality. Thus, matching with the Champion GRL is supplemented by coarse-grained matching. Coarse-grained matching resembles asset reassignment and is utilized only on external characteristics.

3.2.2.1 Asset Set One

Asset Set One contains a list of 10 generalized external asset categories encompassing all external assets. Table 5 provides the external assets comprised within every asset category in Asset Set One.

Table 5: Asset Set One

Asset Category	Assets
DATA_COMPUTATIONAL	DATA_COMPUTATIONAL, DATA_MEMORY, DATA_SENSITIVE
DATA_COMMUNICATION	DATA_COMMUNICATION, DATA_PERIPHERAL
DATA_ENCRYPTION	DATA_ENCRYPTION, KEY
SYSTEM_TIMING	SYSTEM_TIMING, SUBSYSTEM_TIMING

Table 5 (Cont.)

Asset Category	Assets
STATUS	STATUS, READY, DONE, BUSY, HOLD, COUNT, WAIT, COMMUNICATION_STATUS
SYSTEM_CONTROL	SYSTEM_CONTROL, ENABLE, SET, RESET, EXECUTE, READ, WRITE, INTERRUPT, SELECT, HANDSHAKING, SHIFT, LOAD, MODE, INSTRUCTION
ADDRESS_SENSITIVE	ADDRESS_SENSITIVE, REGISTER
SPECIFIC_CONTROL	SPECIFIC_CONTROL, INTERRUPT_CONTROL, PERIPHERAL_CONTROL, REGISTER_FILE_CONTROL, COMMUNICATION_CONTROL, TIMER_CONTROL, CLOCK_CONTROL, COMMUNICATION_PROTOCOL, DATA_OP, MEMORY_OP, INTERRUPT_OP, PROGRAM_COUNTER_OP, BUS_CONTROL, LCD_CONTROL, LED_CONTROL, PHASE, DUTY_CYCLE
EXCEPTION_HANDLING	EXCEPTION_HANDLING, ERROR_HANDLING
EXTRA	EXTRA, CRITICAL, COMPONENT, STATE, UNKNOWN, UNUSED

Two new external assets, *SPECIFIC_CONTROL* and *EXTRA*, were created for coarse-grained asset reassignment, as not all fine-grained assets enjoy a generic equivalent. This version of asset reassignment is used on both the unknown soft IP and the Champion GRL entries to produce the highest possible percentage match between assets and functionality. Table 6 presents an example of asset reassignment using Asset Set One.

Table 6: Example Asset Set One Reassignment

Asset Trace	Original Asset	Reassigned Asset
1	DATA_PERIPHERAL	DATA_COMMUNICATION
2	SYSTEM_TIMING, RESET	SYSTEM_TIMING, SYSTEM_CONTROL

In asset trace 1, the original asset *DATA_PERIPHERAL* is reassigned to *DATA_COMMUNICATION* because *DATA_PERIPHERAL* can also be considered *DATA_COMMUNICATION* since signals labelled as this asset may communicate with other devices. Concerning asset trace 2, the original asset *SYSTEM_TIMING* is unchanged while *RESET* is reassigned to *SYSTEM_CONTROL*. *SYSTEM_TIMING* is the most general asset within the *SYSTEM_TIMING* asset category, so it does not need to be changed. Conversely, *SYSTEM_CONTROL* is the most generic asset which contains *RESET*, resulting in the reassignment above.

During testing, Asset Set One was not able to correctly identify soft IPs with similar functionalities. As an example, the Communication and Peripheral functionalities are similar enough to both be described as Communication respecting Asset Set One. Because of this and the possibility of a soft IP's highest percent Champion GRL match is below a given threshold, a second Asset Set was created.

3.2.2.2 Asset Set Two

Considering how the GRL is defined soft IPs are developed, certain assets, such as *SYSTEM CONTROL* and *TIMING* assets, are more common than any other data asset in the GRL. Keeping this in mind, Asset Set Two classifies each data asset into a new category. Table 7 introduces the external assets of Asset Set Two.

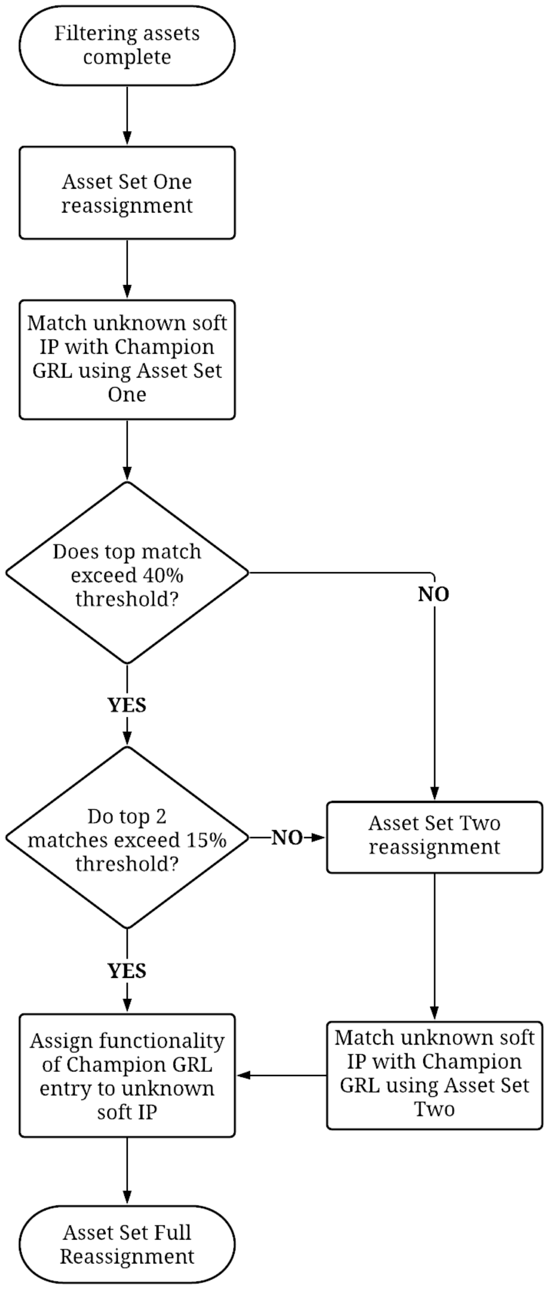
Table 7: Asset Set Two

Asset Category	Assets
DATA_COMPUTATIONAL	DATA_COMPUTATIONAL
DATA_MEMORY	DATA_MEMORY
DATA_COMMUNICATION	DATA_COMMUNICATION
DATA_PERIPHERAL	DATA_PERIPHERAL
DATA_ENCRYPTION	DATA_ENCRYPTION, KEY
DATA_SENSITIVE	DATA_SENSITIVE
SYSTEM_TIMING	SYSTEM_TIMING, SUBSYSTEM_TIMING STATUS, READY, DONE, BUSY, HOLD, COUNT, WAIT, COMMUNICATION_STATUS

Table 7 (Cont.)

Asset Category	Assets
SYSTEM_CONTROL	SYSTEM_CONTROL, ENABLE, SET, RESET, EXECUTE, READ, WRITE, INTERRUPT, SELECT, HANDSHAKING, SHIFT, LOAD, MODE, INSTRUCTION, INTERRUPT_CONTROL, PERIPHERAL_CONTROL, REGISTER_FILE_CONTROL, COMMUNICATION_CONTROL, TIMER_CONTROL, CLOCK_CONTROL, COMMUNICATION_PROTOCOL, DATA_OP, MEMORY_OP, INTERRUPT_OP, PROGRAM_COUNTER_OP, BUS_CONTROL, LCD_CONTROL, LED_CONTROL, PHASE, DUTY_CYCLE, EXCEPTION_HANDLING, ERROR_HANDLING
ADDRESS_SENSITIVE	ADDRESS_SENSITIVE, REGISTER
EXTRA	EXTRA, CRITICAL, COMPONENT, STATE, UNKNOWN, UNUSED

Figure 4 illustrates how to determine if an unknown soft IP requires the use of Asset Set Two or if it can continue matching with designs in its functionality after relying on Asset Set One.



Filtering assets complete
Asset Set One reassignment
Match unknown soft IP with Champion GRL using Asset Set One
Does top match exceed 40% threshold?
NO
YES
Do top 2 matches exceed 15% threshold?
NO
Asset Set Two reassignment
YES
Assign functionality of Champion GRL entry to unknown soft IP
Match unknown soft IP with Champion GRL using Asset Set Two
Asset Set Full Reassignment

Figure 4: Asset Reassignment Flow

Once an unknown soft IP it is subjected to asset reassignment and after these assets have been filtered throughout the soft IP, the improved matching process will automatically reassign assets in compliance with Asset Set One. The matching process will then begin, and the Champion GRL entries will search for a match for the soft IP. The best two matches are output, and the top match is compared with a matching threshold, *40*%. This threshold was determined during initial testing of coarse-grained matching, where it was noted that most unknown soft IPs would give an initial match of above 40%, so any designs with less than a 40% match do not have a high confidence in regards with assigning functionality. If it exceeds *40*%, it is also compared with the second highest matching functionality. This base threshold is used to ensure the design is sorted into the correct functionality and having an unknown soft IP with a highest matching percentage below this threshold is not high enough to trust. A second threshold of *15%* is then used for comparison. This threshold signifies the difference between functionalities and is given this threshold due to the variability in designs within the same and similar functionalities. It was determined during testing of coarse-grained matching using soft IPs that contain similar functionalities, such as *Communication* and *Peripheral* functionalities. If the top 2 matches have a difference larger than *15%*, The unknown soft IP is considered part of the top matches' functionality and matching within the Functionality Golden Reference Library is performed.

3.3 Functionality Golden Reference Library Matching

3.3.1 Functionality Golden Reference Library

As addressed in Section 3.1, the original GRL matched an unknown soft IP against all entries within the library, regardless of functionality. The inclusion of the Champion GRL rendered the original GRL obsolete, so a new GRL was created. This Functionality Golden Reference Library is divided according to whitelisted functionalities defined in Section 3.1. Separating the GRL into functionalities decreases resource demands during the matching process while simultaneously increasing the matching.

3.3.2 Fine-Grained Asset Reassignment

To facilitate GRL entry matching, fine-grained asset reassignment was conceived to increase the matching percentage of soft IPs with Functionality GRL entries. This scheme of asset reassignment contrasts with the others in that only Functionality GRL designs are assigned. Unknown soft IPs used in matching will have the most recent assets assigned to them while Functionality GRL entries may not feature the most up-to-date assets, leading to bias that may negatively affect the matching results. Table 8 shows the list of assets and corresponding asset categories.

Table 8: Asset Set Full

Asset Category	Assets
1	DATA_COMPUTATIONAL, DATA_MEMORY
2	DATA_COMMUNICATION
3	DATA_PERIPHERAL
4	DATA_ENCRYPTION
5	DATA_SENSITIVE
6	SYSTEM_TIMING, SUBSYSTEM_TIMING
7	STATUS, DONE, HOLD, READY
8	BUSY, WAIT
9	COUNT
10	CLOCK_CONTROL, TIMER_CONTROL
11	SET
12	SELECT, ENABLE
13	RESET
14	READ, WRITE, LOAD
15	EXECUTE
16	MODE
17	HANDSHAKING
18	SHIFT
19	INSTRUCTION
20	SYSTEM_CONTROL
21	MEMORY_OP, DATA_OP, REGISTER_FILE_CONTROL

Table 8 (Cont.)

Asset Category	Assets
22	INTERRUPT_OP, INTERRUPT_CONTROL
23	PROGRAM_COUNTER_OP
24	PERIPHERAL_CONTROL
25	COMMUNICATION_CONTROL, COMMUNICATION_PROTOCOL, COMMUNICATION_STATUS
26	INTERRUPT_OP, INTERRUPT_CONTROL
27	CRITICAL
28	COMPONENT
29	ADDRESS_SENSITIVE
30	KEY
31	REGISTER
32	PROGRAM_COUNTER
33	ERROR_HANDLING, EXCEPTION_HANDLING
34	STATE
35	TMS, TCK, TDI, TDO, TRST
36	LCD_CONTROL, LED CONTROL
37	BUS_CONTROL
38	DUTY_CYCLE, PHASE

To establish which asset(s) needs reassignment using the Functionality GRL's specific characteristic, the same characteristic of the unknown soft IP is used. At first, all asset traces from a single characteristic are considered. Only characteristics within external assets are used, as internal assets are automatically assigned during the initial asset assignment step. Next, the process loops through all Functionality GRL entries and receives the asset traces from the same characteristic as the unknown soft IP. Assets from the Functionality GRL are compared against assets within the unknown soft IP. During this step in the matching process, with the most recent and accurate assets assigned to the unknown soft IP, only assets within the GRL are reassigned. If two assets compared from the unknown soft IP and the Functionality GRL entry are the same, no asset reassignment is needed. If the two assets differ but are within the same asset category, the Functionality GRL entry's asset is reassigned to the unknown soft IP's asset.

Table 9: Unknown Soft IP Characteristic

Unknown Soft IP	
Asset Trace	**Asset**
1	DATA_COMMUNICATION
2	COMMUNICATION_PROTOCOL, READY
3	SYSTEM_TIMING
4	ADDRESS_SENSITIVE

Table 10: Functionality GRL Entry Characteristic

Functionality GRL Entry		
Asset Trace	**Original Asset**	**Reassigned Asset**
1	DATA_COMMUNICATION	DATA_COMMUNICATION
2	COMMUNICATION_PROTOCOL, STATUS	COMMUNICATION_PROTOCOL, READY
3	RESET	RESET
4	SUBSYSTEM_TIMING	SYSTEM_TIMING

Together, Tables 9 and 10 provide an example of fine-grained-to-fine-grained asset reassignment. In trace 1, located in Table 10 for the Functionality GRL entry, the *DATA_COMMUNICATION* asset remains the same since *DATA_COMMUNICATION* in Table 9 is the only asset within its category. Trace 2 in Table 10 has one asset that is the same as an asset in the unknown soft IP while the other asset is not. By referencing trace 2 in Table 9 with the unknown soft IP, the asset *STATUS* in asset trace 2 of Table 10 is reassigned to *READY*. Trace 3 of Table 10 does not contain any similar assets with Table 9, so no assets are reassigned. Table 10's asset trace 4, however, does contain a similar asset from Trace 3 in Table 9, so *SUBSYSTEM_TIMING* is reassigned to *SYSTEM_TIMING*.

4. RESULTS AND ANALYSIS

4.1 Champion GRL Results versus Statistical Matching

To confirm the tool's ability to maintain the correct functionality with the changes made, results from [10] were used. The tested IPs include BasicRSA-T200 and RS232-T700. Additional designs from Trust-Hub [11, 12] and OpenCores [13] were used to test the improved matching process.

4.2 Examples

4.2.1 BasicRSA

A Trojan-infested soft IP of BasicRSA was included during testing. It contains a denial-of-service attack which disables encoding at the transmitter and decoding at the receiver.

Table 11: BasicRSA-T200 Matching Results Asset Set One

Target IP	Original Functionality	GRL Entry	Champion GRL Functionality	% Match
RSACypher.vhd	TROJAN_ENCRYPTION_UNIT	RSA-T100	ENCRYPTION_UNIT	97.577%
Modmult.vhd	COMPUTATIONAL	Simple_alu	COMPUTATIONAL	86.877%

Table 11 shows the top result of the BasicRSA-T200 functionalities and their respective percent matches to the given functionalities based on the Champion GRL entry. In both instances, the matching algorithm correctly identifies the functionality of the soft IP using only Asset Set One.

4.2.2 RS232

A Trojan-infested soft IP of RS232 was included during testing. This also contains a denial-of-service attack which targets the transmitter's *done* signal, rendering the transmitter unable to receive communication after completion of a task.

Table 12: RS232-T700 Matching Results Asset Set One

Target IP	Original Functionality	GRL Entry	Champion GRL Functionality	% Match
Uart.vhd	COMMUNICATION	I2c_master	COMMUNICATION	88.837%
U_xmit.vhd	TROJAN_COMMUNICATION	Lcd16x2_ctrl	PERIPHERAL	88.526%
U_rec.vhd	COMMUNICATION	I2c_master	COMMUNICATION	70.972%

Table 13: U_xmit.vhd Matching Results Asset Set One

Target IP	Original Functionality	GRL Entry	Champion GRL Functionality	% Match
U_xmit.vhd	TROJAN_COMMUNICATION	Lcd16x2_ctrl	PERIPHERAL	88.526%
U_xmit.vhd	TROJAN_COMMUNICATION	I2c_master	COMMUNICATION	76.142%

Using only Asset Set One, U_xmit.vhd is classified as having the *Peripheral* functionality in Table 12 with an 88.526% match. However, the entry Uart_xmit.vhd contains a 76.142% match with I2c_master as shown in Table 13. Subtracting the matching percentage of I2c_master from Lcd16x2_ctrl produces 12.384% which is used to determine if Asset Set Two is required. The difference between the two functionalities is within a threshold of 15%; therefore, it is not significant enough to determine the component's own functionality. Due to this, U_xmit had its assets reassigned using Asset Set Two and was compared against the Champion GRL with its assets also reassigned using Asset Set Two.

Table 14: U_xmit.vhd Matching Results Asset Set Two

Asset Set Two		
Champion GRL Functionality	**GRL Entry**	**% Match**
COMMUNICATION	I2c_master	88.716%
PERIPHERAL	Lcd16x2_ctrl	58.657%

Comparing the same two functionalities regarding Asset Set Two, U_xmit.vhd matches closest to the *Communication* functionality. The difference between the two matches increases from 12.384% to 30.059%. Consequently, the highest total match is I2c_master using Asset Set Two, with a total match of 88.716%. All three soft IPs within RS232-T700 are matched to the correct functionality.

4.2.3 PS/2 Keyboard

PS/2 Keyboard was used to ensure that comparisons between the top two matches of Asset Set One were accurate. PS/2 Keyboard is considered a *Peripheral* functionality as it is a device that connects to a computer and is used to communicate a user's keyboard strokes. However, the *Communication* functionality is similar in which components with this functionality also communicate, but the way in which they communicate are different. To further prove the relationship between these two functionalities, PS/2 Keyboard was tested. Table 15 shows the impact of using only Asset Set One to determine functionality.

Table 15: PS/2 Keyboard Matching Results Asset Set One

Target IP	Original Functionality	GRL Entry	Champion GRL Functionality	% Match
Ps2_keyboard.vhd	PERIPHERAL	Lcd_16x2_ctrl	PERIPHERAL	56.792%
Ps2_keyboard.vhd	PERIPHERAL	I2c_master	COMMUNICATION	55.852%

Using only Asset Set One on PS/2 Keyboard, the results show a 56.792% match with the *Peripheral* functionality. In contrast, PS/2 Keyboard matches with the *Communication* functionality at 55.852%. With a difference of less than 1% between the two functionalities, it is difficult to tell if PS/2 Keyboard belongs within the *Peripheral* functionality or the *Communication* functionality based solely on Asset Set One. Asset Set Two is used again to distinguish which functionality best fits this soft IP.

Table 16: PS/2 Keyboard Matching Results Asset Set Two

Asset Set Two		
Champion GRL Functionality	**GRL Entry**	**% Match**
PERIPHERAL	Lcd16x2_ctrl	73.740%
COMMUNICATION	I2c_master	15.228%

After Asset Set Two is used with PS/2 Keyboard, the difference between the *Peripheral* and *Communication* functionalities increase to 58.512%.

4.2.4 Bus Interface

A larger microcontroller was also tested to demonstrate improvements with the new matching process. This microcontroller, named Bus Interface, contains a ROM module, an SPRAM module, LED outputs, and a UART communication module. External assets were assigned to the microcontroller's top module, named Bus_Interface_Top.vhd. External assets were then manually assigned to internal signals. Manual assignment of external assets to internal signals occurs when asset filtering is unable to fully define signals to these subcomponents. After completion of asset assignment and assets were filtered throughout, matching of the microcontroller was performed.

Table 17: Bus Interface Matching Results Asset Set One

Target IP	Original Functionality	GRL Entry	Champion GRL Functionality	% Match
Bus_Interface_Top.vhd	COMMUNICATION	Lcd_16x2_ctrl	PERIPHERAL	49.548%
Osch.vhd	COMMUNICATION	Digi_clock	TIMING	45.833%
PLL_CLK.vhd	TIMING	Shift_8bit	SHIFT_REGISTER	60.273%
Vlo.vhd	COMPUTATIONAL	Data_mem_16	REGISTER_FILE	94.752%
Ehxpllj.vhd	COMMUNICATION	Simple_pic	INTERRUPT_UNIT	63.476%
Bus_Master.vhd	COMMUNICATION	I2c_master	COMMUNICATION	63.304%
SPRAM.vhd	REGISTER_FILE	I2c_master	COMMUNICATION	54.540%
Inv.vhd	COMPUTATIONAL	Simple_alu	COMPUTATIONAL	35.438%
Rom16x1a.vhd	REGISTER_FILE	Decoder2to4	DECODER_ENCODER	59.568%
Vhi.vhd	COMPUTATIONAL	Mc8051_ctrl	CONTROL_GENERATION	84.745%
Fd1p3dx.vhd	CONTROL_GENERATION	Decoder2to4	DECODER_ENCODER	49.658%
Mux321.vhd	COMPUTATIONAL	Decoder2to4	DECODER_ENCODER	87.013%
Spr16x4c.vhd	REGISTER_FILE	Data_mem_16	REGISTER_FILE	85.094%
RS232_Usr_Int.vhd	COMMUNICATION	I2c_master	COMMUNICATION	44.223%
STD_FIFO.vhd	REGISTER_FILE	Lcd_16x2_ctrl	PERIPHERAL	49.789%
Bus_Int.vhd	REGISTER_FILE	I2c_master	COMMUNICATION	60.455%
Std_Counter.vhd	TROJAN_TRIGGER	RSA-T100	ENCRYPTION_UNIT	43.953%
LED_Ctrl.vhd	COMMUNICATION	Lcd_16x2_ctrl	PERIPHERAL	42.536%
PWM_16b.vhd	REGISTER_FILE	Lcd_16x2_ctrl	PERIPHERAL	42.535%

Table 17 shows the results of matching using Asset Set One. Subcomponents that did not have a percentage match high enough to meet the minimum threshold to ensure a functionality had their assets reassigned and are in Table 18.

Table 18: Bus Interface Matching Results Asset Set Two

Target IP	Original Functionality	GRL Entry	Champion GRL Functionality	% Match
Bus_Interface_Top.vhd	COMMUNICATION	I2c_master	COMMUNICATION	35.430%
Vlo.vhd	COMPUTATIONAL	Simple_alu	COMPUTATIONAL	95.628%
Ehxpllj.vhd	COMMUNICATION	RSA-T100	ENCRYPTION_UNIT	78.263%
SPRAM.vhd	REGISTER_FILE	I2c_master	COMMUNICATION	55.025%
Inv.vhd	COMPUTATIONAL	Simple_alu	COMPUTATIONAL	36.980%
Rom16x1a.vhd	REGISTER_FILE	Data_mem_16	REGISTER_FILE	61.424%
Fd1p3dx.vhd	CONTROL_GENERATION	Decoder2to4	DECODER_ENCODER	49.691%
RS232_Usr_Int.vhd	COMMUNICATION	I2c_master	COMMUNICATION	52.616%
STD_FIFO.vhd	REGISTER_FILE	I2c_master	COMMUNICATION	48.318%
Std_Counter.vhd	TROJAN_TRIGGER	Simple_alu	COMPUTATIONAL	64.273%
LED_Ctrl.vhd	COMMUNICATION	I2c_master	COMMUNICATION	55.597%
PWM_16b.vhd	REGISTER_FILE	I2c_master	COMMUNICATION	42.924%

Osch.vhd, PLL_Clock.vhd, Bus_Master,vhd, Vhi.vhd, Mux321.vhd, and Spr16x4c.vhd met both thresholds and did not go through asset reassignment for Asset Set Two. Osch.vhd is an oscillator and can be included in the *Timing* functionality. The statistical matching method incorrectly assigns Osch.vhd to the *Communication* functionality while the new matching method assigns it to the *Timing* functionality. Bus_Master.vhd is another subcomponent within Bus Interface. This component controls the flow of data from within the bus. This design was assigned to *Communication* using both methods. PLL_Clock.vhd is an example of a subcomponent that is assigned the correct functionality when using the statistical method of matching but is assigned to the incorrect functionality using the new method. All other subcomponents and the top-level component were required to use Asset Set Two. Bus_Interface_Top.vhd, Vlo.vhd, Inv.vhd, Rom16x1a.vhd and RS232_Usr_Int all retained the correct functionality assignments using both methods of matching.

Certain subcomponents, such as PLL_Clk.vhd, SPRAM.vhd, and STD_FIFO.vhd were unable to be matched correctly due to current biases with the Champion GRL. Certain functionalities, such as *Timing* and *Control_Generation*, contain entries within the Champion GRL that are smaller than average and less than three asset traces in total, resulting in lower functionality matching results.

4.3 Functionality GRL Results versus Statistical Matching

The same examples from Section 4.2 were used to confirm the correct functionality is assigned to the unknown soft IP as well as to find any discrepancies in fine-grained-to-fine-grained asset reassignment. Differences between total memory resources are also presented to show improvements with the updated matching process not only in accuracy, but in resource management as well.

4.3.1 Basic RSA

Similar to example 4.2.1, comparisons between the statistical matching process and the new matching process to determine the functionality of an unknown soft IP are shown for Basic RSA-T200.

Table 19: BasicRSA-T200 Matching Results

	Statistical Matching Process		New Matching Process	
Target IP	**Functionality**	**% Match**	**Functionality**	**% Match**
RSACypher.vhd	TROJAN_ENCRYPTION_UNIT	83.235%	ENCRYPTION_UNIT	86.557%
Modmult.vhd	COMPUTATIONAL	100%	COMPUTATIONAL	100%

For RSACypher.vhd, the new matching process declares a functionality of *Encryption_Unit* with an 86.557% match, whereas the statistical matching process declares *Trojan_Encrytpciton_Unit* with an 83.235% match. One reason for this discrepancy is due to the

Functionality GRL containing only a handful of encryption units in total, so there are far fewer entries that this soft IP can match with. For Modmult.vhd, the two matching processes both produce the same functionality with a 100% match, respectively. Modmult.vhd is a modular multiplier which is used in encryption units. Entries containing modular multipliers exist within the GRL, resulting in a 100% match using the new matching process.

Table 20: Memory Usage for BasicRSA-T200

Process	Memory Usage
Statistical Matching Process	54 MB
New Matching Process	35 MB

In terms of memory usage, the statistical matching process used 54 Megabytes (MB) while the new process uses only 35, decreasing memory usage by 35%.

4.3.2 RS232-T700

As shown in Table 21, the RS232-T700's functionalities were correctly identified by the statistical and new matching processes. For Uart.vhd, the matching percent for both processes are the same at 100%. U_xmit.vhd and U_rec.vhd, however, contain a lower matching percent using the updated process. The difference in the percent matches between the two demonstrate a bias in favor of the statistical matching process regarding statistical matching as not all designs are located within the same functionality.

Table 21: RS232-T700 Matching Results

	Statistical Matching Process		New Matching Process	
Target IP	**Functionality**	**% Match**	**Functionality**	**% Match**
Uart.vhd	COMMUNICATION	100%	COMMUNICATION	100%
U_xmit.vhd	TROJAN_COMMUNICATION	99.490%	TROJAN_COMMUNICATION	98.806%
U_rec.vhd	COMMUNICATION	94.674%	COMMUNICATION	87.209%

Table 22: Memory Usage for RS232-T700

Process	Memory Usage
Statistical Matching Process	28 MB
New Matching Process	24 MB

Comparing memory usage between the two matching processes, the updated process reduces memory usage by 14%. This slight decrease in memory usage between the statistical matching process and the new process stems from the GRL containing a large number of designs within the *Communication* functionality.

4.3.3 PS/2 Keyboard

Table 23 shows results from both the statistical matching process and the new matching process. Similar to the RS232-T700, PS/2 Keyboard contains the same functionality and the same percent match for both processes.

Table 23: PS/2 Keyboard Matching Results

	Statistical Matching Process		New Matching Process	
Target IP	**Functionality**	**% Match**	**Functionality**	**% Match**
Ps2_keyboard.vhd	PERIPHERAL	100%	PERIPHERAL	100%

Table 24: Memory Usage for PS/2 Keyboard

Process	Memory Usage
Statistical Matching Process	28 MB
New Matching Process	13 MB

Shown in Table 24, memory usage decreased by 53%. This optimization is due to the small number of *Peripheral* oriented designs located within the GRL. A lower number of designs that the Structural Checking tool needs to check decreases the overall matching process time and resource usage.

4.3.4 Bus Interface

Matching results from both processes of matching for Bus Interface are shown in Table 25. Bus_Interface_Top.vhd matched with the *Communication* functionality with a matching percent below 40% in both processes. This is due to this subcomponent pertaining to communication while not being similar enough to designs within the *Communication* functionality. Osch.vhd had a lower matching percent in the new matching process. However, as previously mentioned, Osch belongs in the *Timing* functionality, meaning the new process produced the correct functionality match. The subcomponent Bus_Master matches within its functionality using both matching processes, but the new process generated a nearly 25% higher match.

Table 25: Bus Interface Matching Results

	Statistical Matching Process		**New Matching Process**	
Target IP	**Functionality**	**% Match**	**Functionality**	**% Match**
Bus_Interface_Top.vhd	COMMUNICATION	35.015%	COMMUNICATION	20.563%
Osch.vhd	COMMUNICATION	34.883%	TIMING	28.788%
PLL_CLK.vhd	TIMING	79.183%	SHIFT_REGISTER	70.422%
Vlo.vhd	COMPUTATIONAL	99.157%	COMPUTATIONAL	98.980%
Ehxpllj.vhd	COMMUNICATION	52.908%	ENCRYPTION_UNIT	53.571%
Bus_Master.vhd	COMMUNICATION	69.459%	COMMUNICATION	87.659%
SPRAM.vhd	REGISTER_FILE	91.698%	COMMUNICATION	84.028%
Inv.vhd	COMPUTATIONAL	78.355%	COMPUTATIONAL	62.422%
Rom16x1a.vhd	REGISTER_FILE	66.772%	REGISTER_FILE	65.261%
Vhi.vhd	COMPUTATIONAL	95.520%	CONTROL_GENERATION	95.693%
Fd1p3dx.vhd	CONTROL_GENERATION	69.512%	DECODER_ENCODER	51.471%
Mux321.vhd	COMPUTATIONAL	61.335%	DECODER_ENCODER	96.598%
Spr16x4c.vhd	REGISTER_FILE	94.439%	REGISTER_FILE	94.265%
RS232_Usr_Int.vhd	COMMUNICATION	69.639%	COMMUNICATION	44.223%
STD_FIFO.vhd	REGISTER_FILE	66.154%	COMMUNICATION	79.112%
Bus_Int.vhd	REGISTER_FILE	74.852%	COMMUNICATION	74.666%
Std_Counter.vhd	TROJAN_TRIGGER	54.798%	COMPUTATIONAL	79.893%
LED_Ctrl.vhd	COMMUNICATION	60.795%	COMMUNICATION	49.970%
PWM_16b.vhd	REGISTER_FILE	67.058%	COMMUNICATION	62.868%

Overall, microprocessors and other large designs that contain multiple subcomponents are difficult to classify using the GRL at this time. This is due to the large number of assets that can be assigned to a signal and a small number of microprocessors and controller entries within the GRL.

Table 26: Memory Usage for Bus Interface

Process	**Memory Usage**
Statistical Matching Process	239 MB
New Matching Process	54 MB

Table 26 shows the impact of this new matching process on the Bus Interface in terms of memory usage. The new matching process decreases memory usage by 77%, using only 54 MB in total. This is in contrast with the statistical matching process's 239 MB used. The significant decrease in memory results from the large number of sub-level entries within the GRL that the statistical process had to check.

5. CONCLUSION AND FUTURE WORK

Improvements in asset reassignment and the creation of the Champion Golden Reference Library and the Functionality Golden Reference Library enhanced the efficiency of the matching process for the Structural Checking tool while also maintaining a high level of accuracy regarding functionality matching. By including a subset of Golden Reference Library entries to match an unknown soft IP, memory is saved by up to 77%. Unknown soft IPs that have similar functionalities can be distinguished when using multiple Asset Sets with relative accuracy. The fluctuations in matches using Asset Set Full indicate reassigning assets may change the overall functionality of an unknown soft IP. One Trojan-infested soft IP changed from *Trojan_Encryption_Unit* to *Encryption_Unit* with the difference between the two functionalities being 5.24%. Microcontrollers are an example of soft IP that have a relatively low matching percentage due to the limited number of entries the GRL contains. Future work can continue to grow the list of functionalities as well as improve designs within the Champion GRL to decrease the use of Asset Set Two. The addition of new external assets can benefit the new functionality matching process by more effectively classifying unknown soft IPs.

6. REFERENCES

[1] M. Tehranipoor and C. Wang. 2012. *Introduction to Hardware Security and Trust*. Springer.

[2] L. N. Nguyen, C. Cheng, M. Prvulovic and A. Zajić, "Creating a Backscattering Side Channel to Enable Detection of Dormant Hardware Trojans," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 7, pp. 1561-1574, July 2019, doi: 10.1109/TVLSI.2019.2906547.

[3] D. Qiang, S. Lisong, S. Qiang, C. Jihua, Y. Daheng and L. Jun, "Hardware Trojan detection by differential side-channel analysis (DDSA)," *2020 IEEE 5th International Conference on Signal and Image Processing (ICSIP)*, Nanjing, China, 2020, pp. 1021-1026, doi: 10.1109/ICSIP49896.2020.9339301.

[4] H. Shen, H. Tan, H. Li, F. Zhang and X. Li, "LMDet: A "Naturalness" Statistical Method for Hardware Trojan Detection," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 4, pp. 720-732, April 2018, doi: 10.1109/TVLSI.2017.2781423.

[5] C. H. Kok *et al*., "Net Classification Based on Testability and Netlist Structural Features for Hardware Trojan Detection," *2019 IEEE 28th Asian Test Symposium (ATS)*, Kolkata, India, 2019, pp. 105-1055, doi: 10.1109/ATS47505.2019.00020.

[6] Weaver, L., Le, T., & Di, J. (2016). Golden Reference Library Matching of Structural Checking for securing soft IPs. *SoutheastCon 2016*. doi:10.1109/secon.2016.7506737

[7] M. Hinds, J. Brady, and J. Di, "Signal Assets - a Useful Concept for Abstracting Circuit Functionality," presented at the Government Microcircuit Applications & Critical Technology Conference (GOMACTech), 2013.

[8] T. Le, J. Di, M. Tehranipoor, and L. Wang, "Tracking data flow at gate-level through structural," in *2016 International Great Lakes Symposium on VLSI (GLSVLSI)*, 2016, pp. 185-189.

[9] J. Yust, M. Hinds, and J. Di, "Structural Checking: Detecting Malicious Logic without a Golden Reference," *Journal of Computational Intelligence and Electronic Systems,* vol. 1, no. 2, p. 8, 2012.

[10] B. McGeehan, F. Smith, T. Le, H. Nauman and J. Di, "Hardware IP Classification through Weighted Characteristics," *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, Waltham, MA, USA, 2019, pp. 1-6, doi: 10.1109/HPEC.2019.8916225

[11] H. Salmani, M. Tehranipoor, and R. Karri, "On Design vulnerability analysis and trust benchmark development", IEEE Int. Conference on Computer Design (ICCD), 2013.

[12] B. Shakya, T. He, H. Salmani, D. Forte, S. Bhunia, M. Tehranipoor, "Benchmarking of Hardware Trojans and Maliciously Affected Circuits", Journal of Hardware and Systems Security (HaSS), April 2017.

[13] *OpenCores*. Available: http://opencores.org/