國立臺灣大學電機資訊學院電子工程學研究所

碩士論文

Graduate Institute of Electronics Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

# 考量木馬插入和前端探測攻擊之實體設計

# Security-aware Physical Design against Trojan Insertion, Frontside Probing, and Fault Injection Attacks

許致瑋

Jhih-Wei Hsu

指導教授：張耀文博士

Advisor: Yao-Wen Chang, Ph.D.

中華民國 111 年 7 月

July 2022

doi:10.6342/NTU202202298

# 國立臺灣大學碩士學位論文
# 口試委員會審定書

考量木馬插入和前端探測攻擊之實體設計
Security-aware Physical Design against Trojan Insertion, Frontside Probing, and Fault Injection Attacks

本論文係許致瑋君（R09943095）在國立臺灣大學電子工程學研究所完成之碩士學位論文，於民國 111 年 07 月 26 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

張 耀 文
（指導教授）

[illegible] 張原豪

[illegible] 江蕙如

系主任、所長 江介宏

# Acknowledgements

I would like to express my gratitude to my advisor, Professor Yao-Wen Chang, for his tremendous advice, instruction, and guidance throughout my pursuit of the Master's degree. Besides, I greatly appreciate the committee members of my oral defense, Professor Hui-Ru Jiang, Professor Ting-Ting Hwang, Professor Yuan-Hao Chang, and Professor Peter B. Luh for their valuable comments and suggestions.

I am grateful to every member of the Electronic Design Automation Laboratory in National Taiwan University for many inspiring discussions and useful comments on my research. Especially, I would like to thank Kuan-Cheng Chen, Yan-Syuan Chen, Yu-Hsiang Lo, Cheng-Yuan Wang, Yen-Ting Chen, Cheng-Yu Chiang, Yu-Ching Huang, Chien-Hao Tsou, and Wei-Hsiang Tseng for their friendship and assistance with my research.

My deepest appreciation goes to my dear parents. Without their constant support and endless love, the completion of this thesis would not have been possible.

Jhih-Wei Hsu

*National Taiwan University*
*July 2022*

doi:10.6342/NTU202202298

考量木馬插入和前端探測攻擊之實體設計

研究生：許致瑋　　　指導教授：張耀文 博士

國立臺灣大學電子工程學研究所

## 摘要

硬體攻擊事件的急速增長以及設計工具缺乏安全性考量的解決方案造成嚴重的安全性問題在現代的晶片設計中。雖然有許多現有的對策對安全性的議題提供不錯的保護，但這些方法仍然缺乏在設計階段有充分考量安全性問題的全局觀點(Global design view)。本論文提出了一個在設計階段考量木馬插入(Trojan insertion)、前端探測(Frontside probing)和故障注入(Fault injection)攻擊之安全性框架。該框架主要由兩大技術組成：(1) 能有效覆蓋敏感物件之暴露面積的大規模屏蔽方法 (2) 基於單元元件移動的方法來消除易受木馬插入的空間。

實驗結果顯示，我們的框架能有效的抵擋這些攻擊的漏洞，並在與 2022

doi:10.6342/NTU202202298

年 ISPD 安全性收斂之實體佈局競賽(2022 ACM ISPD Security Closure of Physical Layouts Contest)的前三名隊伍相比，能取得最佳的總分。

**關鍵詞**：實體設計、木馬插入、前端探測攻擊、故障注入攻擊

doi:10.6342/NTU202202298

# SECURITY-AWARE PHYSICAL DESIGN AGAINST TROJAN INSERTION, FRONTSIDE PROBING, AND FAULT INJECTION ATTACKS

**Student: Jhih-Wei Hsu  Advisor: Dr. Yao-Wen Chang**

**Graduate Institute of Electronics Engineering**
**National Taiwan University**

## Abstract

The dramatic growth of hardware attacks and the lack of security-concern solutions in design tools lead to severe security problems in modern IC designs. Although many existing countermeasures provide decent protection against security issues, they still lack the global design view with sufficient security consideration in design time. This thesis proposes a security-aware framework against Trojan insertion, frontside probing, and fault injection attacks at the design stage. The framework consists of two major techniques: (1) a large-scale shielding method that effectively covers the exposed areas of assets and (2) a cell-movement-based method to eliminate the empty spaces vulnerable to Trojan insertion.

Experimental results show that our framework effectively reduces the vulnerability of these attacks and achieves the best overall score compared with the top-3 teams in the 2022 ACM ISPD Security Closure of Physical Layouts Contest.

**Keywords**: Physical Design, Trojan Insertion, Frontside Probing Attacks, Fault Injection Attacks

doi:10.6342/NTU202202298

# Table of Contents

doi:10.6342/NTU202202298

doi:10.6342/NTU202202298

# List of Tables

doi:10.6342/NTU202202298

# List of Figures

doi:10.6342/NTU202202298

doi:10.6342/NTU202202298

# Chapter 1

# Introduction

In this thesis, a security-aware physical design framework for security- aware layout design problem is proposed. In the following section, we first introduce security-aware layout design problem in Section 1.1. Next, a survey of related works is given in Section 1.2. Then, we explain our motivation for dealing with the problem in Section 1.3. After that, we list our contributions to this thesis in Section 1.4. Finally, Section 1.5 shows the organization of the rest of this thesis.

## 1.1 Introduction to a Security-aware Layout Design Problem

The integrated circuit (IC) fabrication process has become more and more complicated as technology advances. As a result, IC companies started outsourcing the design and production of the chips to lower the manufacturing costs. As the semiconductor fabrication process requires many stages from upstream to downstream, attackers could have plenty of opportunities to modify ICs maliciously [13]. Figure 1.1 shows different security threats during the life-cycle of ICs [13]. Besides, ICs are applied to many applications, such as mobile phones, communication, transportation, and other critical domains. If the ICs are successfully attacked, it could cost great losses to the semiconductor industry and even jeopardise civilians' lives. Consequently, these issues have raised serious concerns about the authenticity of the

doi:10.6342/NTU202202298

fabrication process and the trustworthiness of manufactured ICs. The demand for hardware defense methods is getting more critical than before.

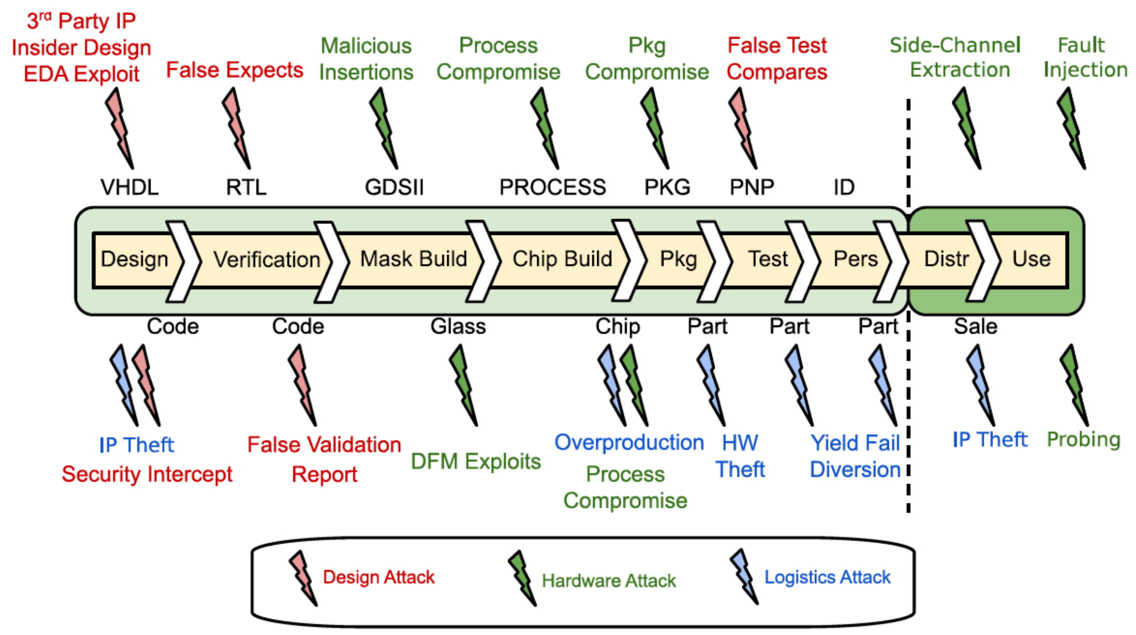


Figure 1.1: Different security threats during the life-cycle of ICs [13].

There are two straightforward methods to maintain the reliability of ICs. First, we could thoroughly inspect whether the manufactured chips are damaged before selling them. Nevertheless, this method is not effective enough since ICs are not repairable after fabrication. Second, we could make sure that all the stages in the supply chain are trustworthy. However, this solution is difficult and expensive since the suppliers are all around the globe [21].

For the above reasons, *secure-by-design* has gradually drawn attention. Secure-by-design is a paradigm indicating that confidentiality, integrity, and availability must be considered throughout the computer-aided design (CAD) flow, from the specification stage to the fabrication and assembly stages [13]. The objective of secure-by-design is to proactively protect the ICs and ensure the effects of secu-

doi:10.6342/NTU202202298

rity methods implemented in the early design stages could remain consistent in the following stages.

*Trojan insertion*, *frontside probing*, and *fault injection attacks* are three most common hardware attacks [13]. Trojan insertion, also known as hardware Trojan attacks, is a malicious modification or inclusion made by suspicious third parties [28]. An IC attacked by Trojan insertion might change its original behavior, leak private information, or even the whole circuitry could be permanently damaged, thereby losing its functionality. Besides, the types of hardware Trojans are of great variety, and they are becoming harder to be detected since the attacking techniques have been advancing these years [28]. Therefore, the threats brought by Trojan insertion are tremendous. As shown in Figure 1.2, general layouts without implementing any defense method would have numerous exploitable regions, which are the continuous empty spaces vulnerable to insert Trojans. In previous research, Tehranipoor *et al.* [21] revealed the vulnerability of ICs against malicious Trojan attacks. In view of the reasons mentioned above, it is extremely urgent to develop effective Trojan-resisting countermeasures.

doi:10.6342/NTU202202298

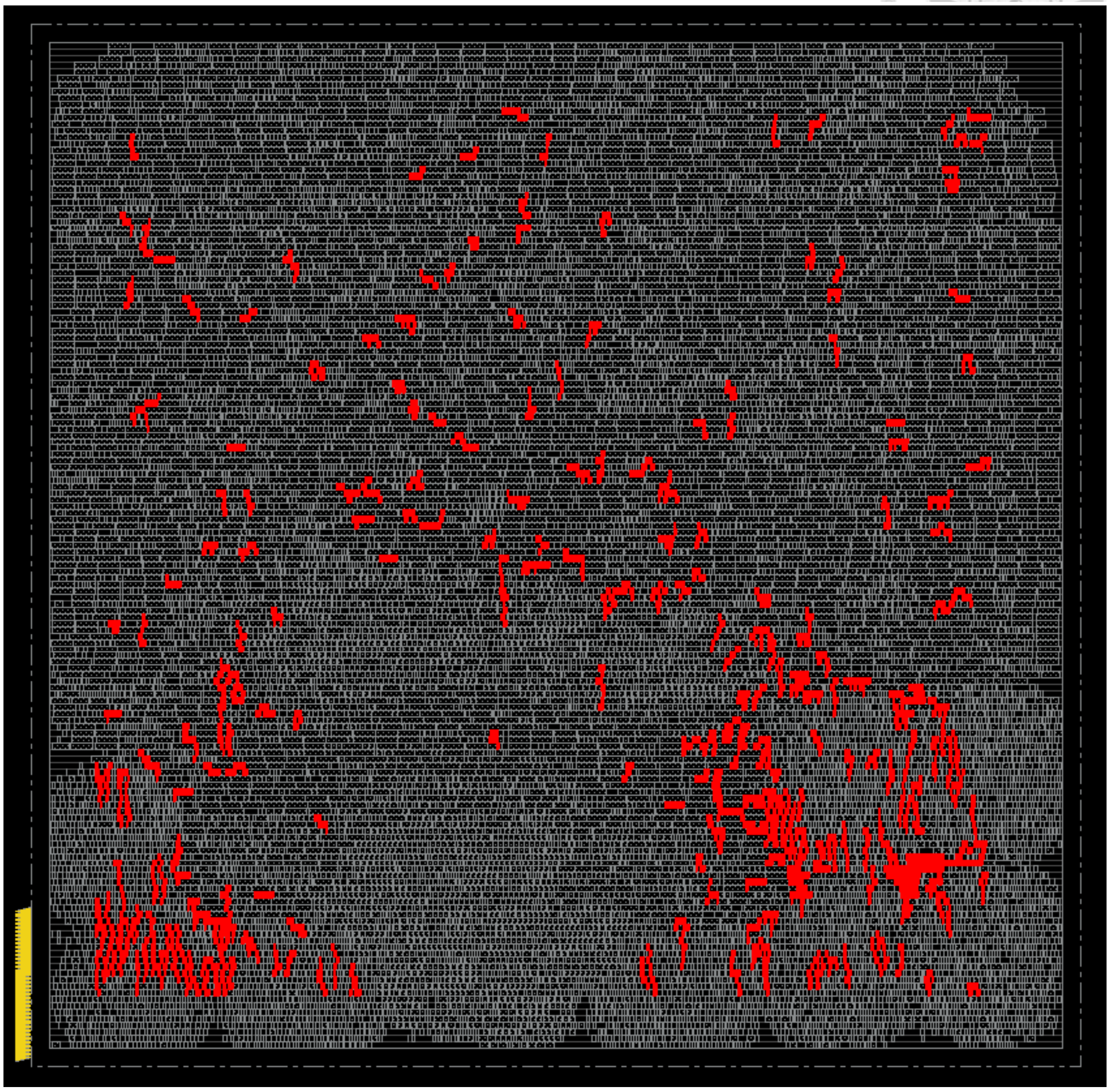


Figure 1.2: An advanced encryption standard (AES) layout without implementing any defense method against Trojan insertion. The red marks are the exploitable regions.

Probing attacks are invasive attacks meant to leak the private information of the ICs. As shown in Figure 1.3, probing attacks would bypass the security measures and expose the cells and nets containing sensitive information in the beginning. Afterward, attackers would use hardware tools to physically attach to the cells and nets exposed in the previous step and extract information from them [25]. At the application level, probing attacks could be implemented on security-critical IC devices, such as smart cards, smartphones, financial systems, and even military systems [26], which also indicates the significant risks to confidential information brought by probing attacks.

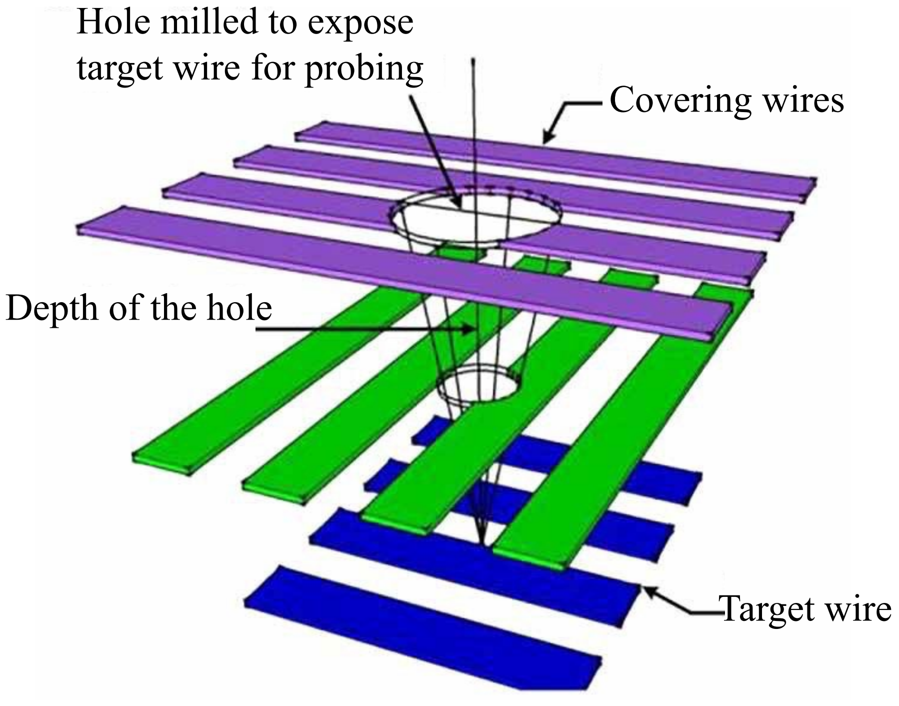


Figure 1.3: Illustration of the frontside probing attacks with focused ion beams (FIBs) [25].

doi:10.6342/NTU202202298

Fault injection attacks (FIAs) have also drawn attention in the last decade, and it has been proven to be highly effective for private hardware information leaking [3]. FIA aims at physically interfering with the ICs beyond its original functionality, and the errors or side-effects induced could be used to crack cryptographic keys and other secret data [20]. Using FIA and analyzing the abnormal outputs of the ICs, which is called *differential fault analysis (DFA)*, could significantly reduce the number of experiments needed originally for obtaining the secret keys [3]. Besides, FIA is a relatively new attack method, so it has a great potential to become more powerful in the near future, which also shows an urgent need for FIA countermeasures.

This thesis aims to create secure-by-design countermeasures to cope with Trojan insertion, frontside probing, and fault injection attacks.

doi:10.6342/NTU202202298

## 1.2 Related Works

In this section, we introduce some of the previous countermeasures that address the three common attacks mentioned above. The countermeasures are split into two branches that are given in Section 1.2.1 and Section 1.2.2.

### 1.2.1 Trojan Insertion Countermeasures

Wang *et al.* [27] presented the complexity and strength of Trojans, revealing that it is challenging to detect them after they are inserted into the hardware. Xiao *et al.* [28] proposed three classes of Trojan prevention methods, including (1) logic obfuscation, (2) camouflaging, and (3) functional filler cells. The concept of logic obfuscation is to hide the original implementation or functions of a design by inserting built-in locking mechanisms into it. The camouflaging technique aims at confusing the attackers by adding fake contacts and connections between nodes. Last, functional filler cells aim at filling unused empty spaces with dummy modules so that there is no room for Trojans [28].

Xiao *et al.* [29] present a novel technique called built-in self-authentication (BISA) that makes Trojan insertion by untrusted third-party developers or untrusted foundry more complex and easier to detect. BISA eliminates empty spaces and replaces them with functional filler cells instead of nonfunctional filler cells. Nabeel *et al.* [17] proposed an active interposer that can integrate untrusted chiplets securely at the system level. It is an active method that monitors malicious behavior at runtime. Guo *et al.* [7] proposed a new language-based framework called QIF-Verilog. The QIF-Verilog framework presents a quantified information flow (QIF) model to evaluate the trustworthiness of a hardware system at the register transfer level (RTL). [11] and [15] proposed detection-based methods for detecting hardware Trojans during design, manufacturing, and testing.

doi:10.6342/NTU202202298

### 1.2.2 Probing Attack and Fault Injection Attack Countermeasures

Cioranesco *et al.* [6] proposed an active shield method that detects milling by creating wires carrying dynamic signals on the top metal layer as a protective shield. Wang *et al.* [26] proposed a focused ion beam (FIB)-aware anti-probing physical design flow to protect the crucial wires by internal shield nets. The techniques aim at routing the non-security-related nets above the target nets to create a shield.

Homma *et al.* [8] proposed a sensor-based method to defend against electromagnetic (EM) analysis and fault injection attacks. An EM attack sensor leverages LC oscillators to detect variations in the EM field around a cryptographic large-scale integration (LSI) induced by a tiny probe brought near the LSI. Ngo *et al.* [18] proposed a new shield structure called "cryptographically secure shield" against probing attacks. The cryptographically secure shield architecture comprises (1) the control part and (2) the shield mesh. The control part generates and checks the random bits circulating through the mesh. The shield mesh consists of multiple nets on the top-most metal layer. Khairallah *et al.* [12] proposed the first differential fault analysis-aware physical design automation approach that effectively alleviates the threat posed by differential fault analysis (DFA). DFA is a sophisticated mathematical analysis technique that can precise fault injection attacks. They developed a novel floorplan heuristic and a routing mechanism that integrated a ring oscillator-based sensor circuit near the potential attack targets.

doi:10.6342/NTU202202298

## 1.3 Motivation

Most hardware defense methods, especially Trojan insertion, are detection-based [7,11,15,17,28]. However, hardening the layouts against attacks at the design stage would be more secure. Current Trojan insertion countermeasures focus on detecting Trojans, and these detection methods could be vulnerable when the Trojan insertion techniques advance. Besides, current frontside probing attack and fault injection attack countermeasures cannot remove all the exposed areas and fully protect the cell and net assets. Furthermore, most previous works mainly focus on security and neglect the rise in design costs brought by their countermeasures.

doi:10.6342/NTU202202298

## 1.4 Our Contributions

This thesis proposes a framework to proactively harden the layout against Trojan insertion, frontside probing, and fault injection attacks. Unlike previous countermeasures that attempted to identify and block hardware attacks, our flow focuses on building a security-aware layout. The detailed contributions are summarized below:

- We propose an effective defense framework against Trojan insertion, frontside probing, and fault injection attacks. The framework consists of our proposed pre-processing techniques with two primary defense methods, (1) a large-scale shielding method that could effectively remove the exposed areas and (2) an effective cell movement algorithm to eliminate the exploitable regions.
- We propose pre-processing techniques and integrate them into the conventional physical design flow. As a result, we can create a suitable placement and routing result for our large-scale shielding method.
- We propose a large-scale shielding method that can effectively protect all the cell and net assets from frontside probing attacks.
- We propose a cell movement algorithm at the post-route stage, which could effectively eliminate exploitable regions while restraining the power cost.
- Our proposed framework also considers comprehensive design costs, including power, timing, and area, which is not often considered in previous defense methods.
- Experiment results show that our proposed framework can significantly reduce exploitable regions and exposed areas. Even more, we achieve the best scores

among the participating teams at the 2022 ACM ISPD Security Closure of Physical Layouts contest [2], which justifies the effectiveness of our work.

## 1.5 Thesis Organization

The remainder of this thesis is organized as follows. Section 2 introduces the addressed attack methods, describes the evaluation models, and formulates the security-aware layout design problem. Section 3 details our proposed framework flow and the techniques used at each stage. Section 4 reports the experiment results, and Section 5 concludes this thesis.

doi:10.6342/NTU202202298

# Chapter 2

# Preliminaries

In this chapter, we first introduce three common attacks in Section 2.1, and Section 2.2. Second, we give the evaluation methods used in this thesis in Section 2.3. Third, we give our terminologies and notations used in this thesis in Section 2.4. Finally, we give our problem formulation of the security-aware design framework for Trojan insertion, frontside probing, and fault injection attacks.

## 2.1 Trojan Insertion Attack

Trojan insertion attacks are a malicious hardware modification of an electronic circuit, which leads to incorrect behavior during operation [4]. The concept of Trojans is diverse, including those (1) targeting at the system level, behavior level, gate level, or physical level, (2) intending to leak sensitive information or destroy the functionality of the circuit, and (3) crafted by additive, substitution, or subtractive techniques [13]. In this thesis, we focus on post-design time gate-level addictive Trojan attacks.

Most Trojans comprise a trigger and a payload, as shown in Figure 2.1. The trigger is an optional part that monitors the circuit signals and activates the payload when an expected event occurs. The payload performs the actual attack when receiving the signal from the trigger. Otherwise, the payload remains inert,

doi:10.6342/NTU202202298

and the circuit performs as a Trojan-free circuit [28].

Trippel *et al.* [23] mentioned that a successful Trojan attack requires all the three conditions: (1) Trojan placement, a spatial space to place the additional Trojan components, (2) Trojan integration, a connection between Trojan payload and security components, and (3) Intra-Trojan routing, a connection among the trigger and payload portions of Trojan. To complete a Trojan insertion attack, the attackers need at least a spatial space along with routing resources to wire up the Trojan. Thus, the evaluation method in Section 2.3.1 will be based on this observation.

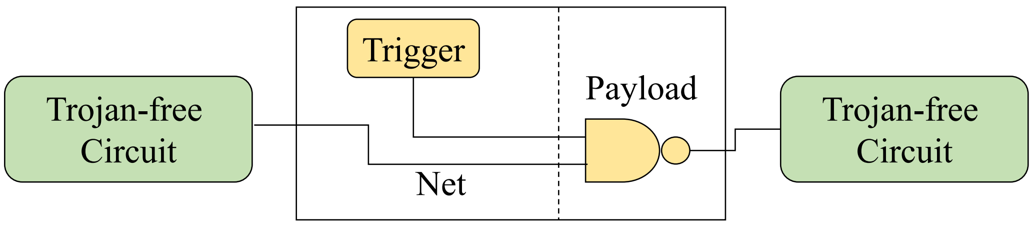


Figure 2.1: A Trojan consists of a trigger and a payload. To perform a Trojan attack, intra-Trojan routing and Trojan integration are required. That is, the trigger should be connected to the payload as well as a Trojan-free target asset.

## 2.2 Frontside Probing and Fault Injection Attack

A probing attack is an invasive physical attack that enables the probe to expose the cells or nets to extract sensitive information through the frontside metal layers or the backside substrate [13]. The concept of probing attacks is diverse and covers those (1) using contact-based micro-probing, electromagnetic field probing, or electro-optical device probing to achieve an attack [2] and (2) aiming to obtain on-chip information, acquire device configuration, or destroy system functionality [13].

This thesis focuses on the frontside contact-based micro-probing attack, which requires direct access to the target cells or wires and is often achieved by using techniques like the focused ion beam (FIB) [2]. First, an FIB mills a cavity with an ion beam to access the targeted wires, as shown in Figure 2.2(a). Then, some metal gas atoms are injected and can be deposited in the cavity to build a conducting path called an electrical probe contact, as shown in Figure 2.2(b). Last, the attackers can extract the asset signal via the electrical probe contact [26]. Note that all the steps above should not damage the upper layer circuitry in order not to destroy the functions of the ICs.

Fault injection attacks aim to deduce sensitive information by directly or indirectly injecting faults during the cryptographic operation. Direct fault injection, for example, can be completed by using laser light or electromagnetic waves. On the other hand, indirect fault injection can be performed by repetitively writing to particular memory locations or by deliberating misuse of dynamic voltage and frequency scaling (DVFS) features [2].

This paper concentrates on frontside direct fault injection, like laser fault injection. First, the laser light sneaking through the metal stacks directs to the target assets. Due to the photoelectric effect, electron-hole pairs are created and

doi:10.6342/NTU202202298

then induce the transient current. Finally, this may affect the PN junction and the switching of transistors of the related cells [2].

To sum up, the frontside probing and fault injection attacks addressed in this paper share the same attack principle, aiming to get down to the target assets via the metal stack. Thus, the evaluation method in Section 2.3.2 will be based on this observation.

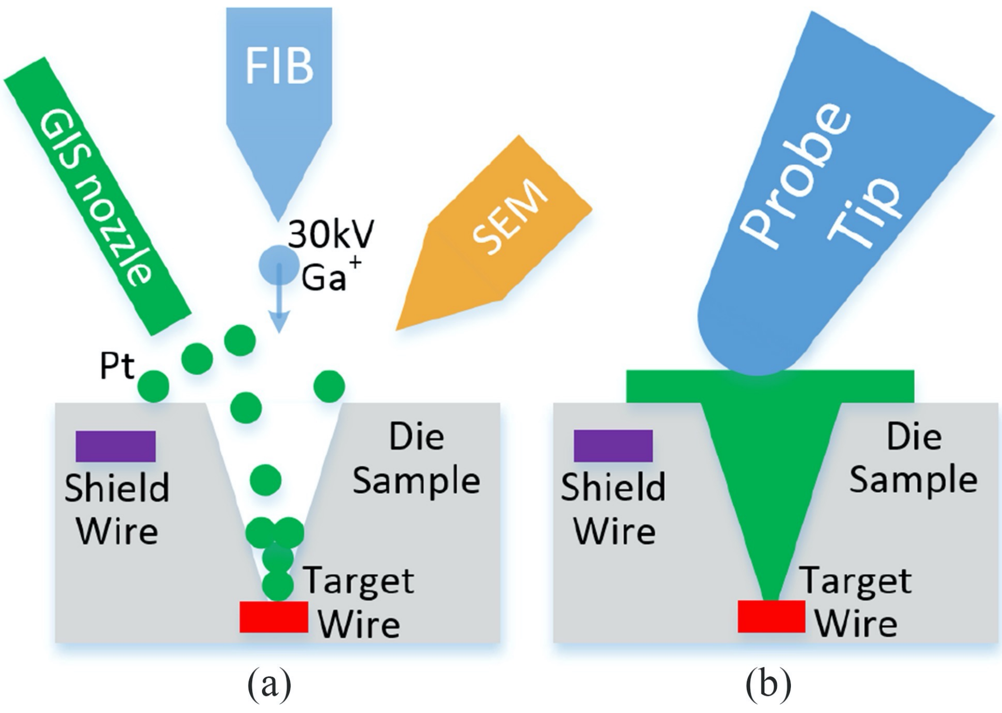


Figure 2.2: Illustrations of a focused ion beam (FIB). (a) An FIB mills a cavity to reach the target wire. (b) The deposition of the metal gases forms the conducting path which serves as the electrical probe contact [26].

doi:10.6342/NTU202202298

## 2.3 Evaluation

The evaluation methods in this thesis follow the 2022 ACM ISPD Security Closure of Physical Layouts Contest [2]. This section introduces the exploitable region for Trojan insertion and the exposed area for frontside probing and fault injection attacks.

### 2.3.1 Exploitable Region for Trojan Insertion Attacks

An *exploitable region* is a region that can be used by attackers to insert Trojans while meeting the three requirements mentioned in Section 2.1. Specifically, it can be an empty region that is big enough to accommodate the Trojan components and have sufficient routing resources to route the Trojan.

The paper defines exploitable regions as the sets of spatially continuous sites larger than the Trojan-placement threshold, which is the minimal exploitable placement sites. To estimate the severity of the threat, we compute the maximum number, the average number, and the total number of the sites across all exploitable regions. The three values can help evaluate the exploitable regions more comprehensively. The routing resource of the exploitable regions is defined by the number of free tracks above the exploitable regions. To simplify the evaluation process, we evaluate the number of all tracks above the exploitable regions instead.

In addition, Trojans should be placed and routed at appropriate positions so that the timing constraints of the design are satisfied. Thus, an exploitable region is only defined within the exploitable distance of cell assets. The exploitable distance is the farthest distance among all legal Trojan-placement positions from the target assets. The exploitable distance is computed as follows. First, we extract paths related to the assets with positive timing slacks. Then, we virtually insert an additional Trojan circuit. Without loss of generality, we use NAND gates, the most simple form of Trojan, for evaluation. Next, we estimate the delays resulting from the additional Trojan circuit. For simplification, we do not perform actual routing. Instead, we estimate the slacks with information from the library, like wiring loads and capacitance loads. Finally, we determine the exploitable distance as the maximal distance with at least one path of positive slacks.

To sum up, we determine the exploitable regions as the sufficient continuous spatial spaces within the exploitable distance and further evaluate the Trojan insertion attack with the exploitable region and the routing resources above the exploitable region.

### 2.3.2 Exposed Area for Frontside Probing and Fault Injection Attacks

An *exposed area* is the region of the cell and net assets accessible by the probes from the frontside via the metal layers. Same as the ISPD contest, we assume that the probes are infinitely thin and only considers the attack probing from the frontside at zero degrees for simplification. That is, we do not consider the capabilities of adversaries and compute all the regions accessible through a direct line from the right top. This assumption is reasonable because some fault injection techniques do not need a large exposed area to induce fault injection. For example, laser fault injection requires just a small transistor-sized exposed area to induce fault injection. Meanwhile, the increasing exposed area provides an expanding attack surface, so the attacker gains more chances to induce fault injection on the target assets. Hence, we conclude that the exposed area is positively related to the security threat of assets. Here we define the exposed area as any region of the cell and net assets visible from the frontside by a direct line through the metal stacks.

doi:10.6342/NTU202202298

Figure 2.3 shows an example of the exposed area of the standard cells in a layout. For evaluation, we further compute the maximum percentage, average percentage, and total of exposed area across all assets to cover the different aspects of the threat from frontside probing and fault injection attacks.

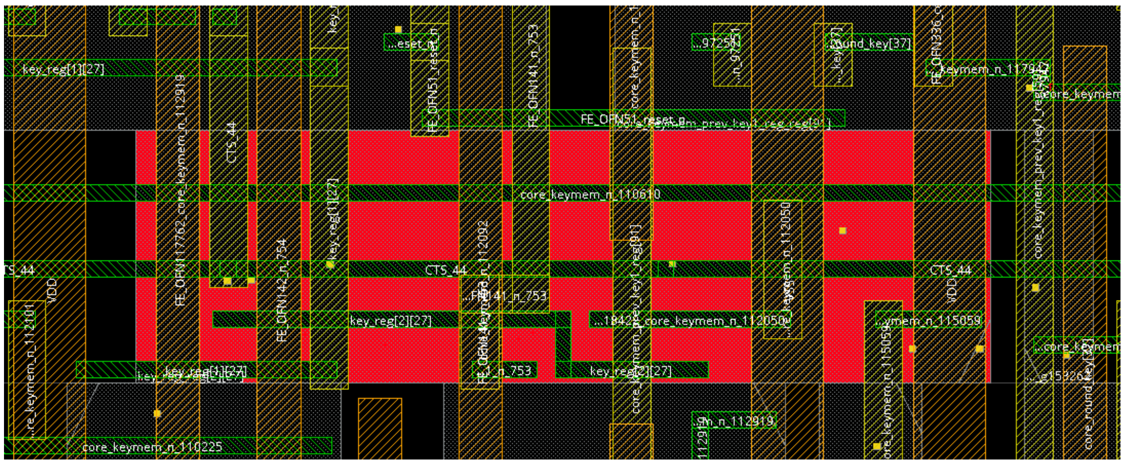


Figure 2.3: An example of the exposed area of a cell. The exposed area is marked in red.

## 2.4 Terminologies and Notations

Here, we give some notations used throughout this thesis:

- $G_a$ is a asset group.
- $m$ is a margin.
- $\alpha_x$ and $\alpha_y$ are two user-defined parameters to adjust the position of the region of a asset group.
- $L_c$ is a cell assets list.
- $d_e$ is a exploitable distance.

- $t$ is a Trojan-placement threshold.
- $S_e$ is a set of exploitable region list.
- $G$ is a row-based graph.
- $c_i$ is a standard cell $i$.
- $n_i$ is a row-based sub-region $i$.

## 2.5 Problem Formulation

The problem formulated in this thesis follows the 2022 ACM ISPD Security Closure of Physical Layouts Contest [2]. The security-aware layout design problem for Trojan insertion, frontside probing, and fault injection attacks is formally defined as follows:

**Problem 1** (**The Security-aware Layout Design Problem for Trojan Insertion, Frontside Probing, and Fault Injection Attacks**). *Given a cell assets list, a net assets list, and a netlist of an original design, determine the desired position for each cell and net subject to the following hard constraints so that the security threat posed by Trojan insertion, frontside probing, and fault injection attacks can be minimized.*

- Must maintain functional equivalence.
- Must maintain cell and net assets.
- Must maintain the ratio of area covered by power delivery network(PDN) to die area relatively constant; variations of +/- 10% in area coverage are permitted.

- Cannot add extra metal layers.
- Cannot add customized, specialized circuitry.
- Cannot relocate PDN to a different layer.
- Cannot create custom cells, meaning that only cells defined in the given LEF files can be used.

doi:10.6342/NTU202202298

# Chapter 3

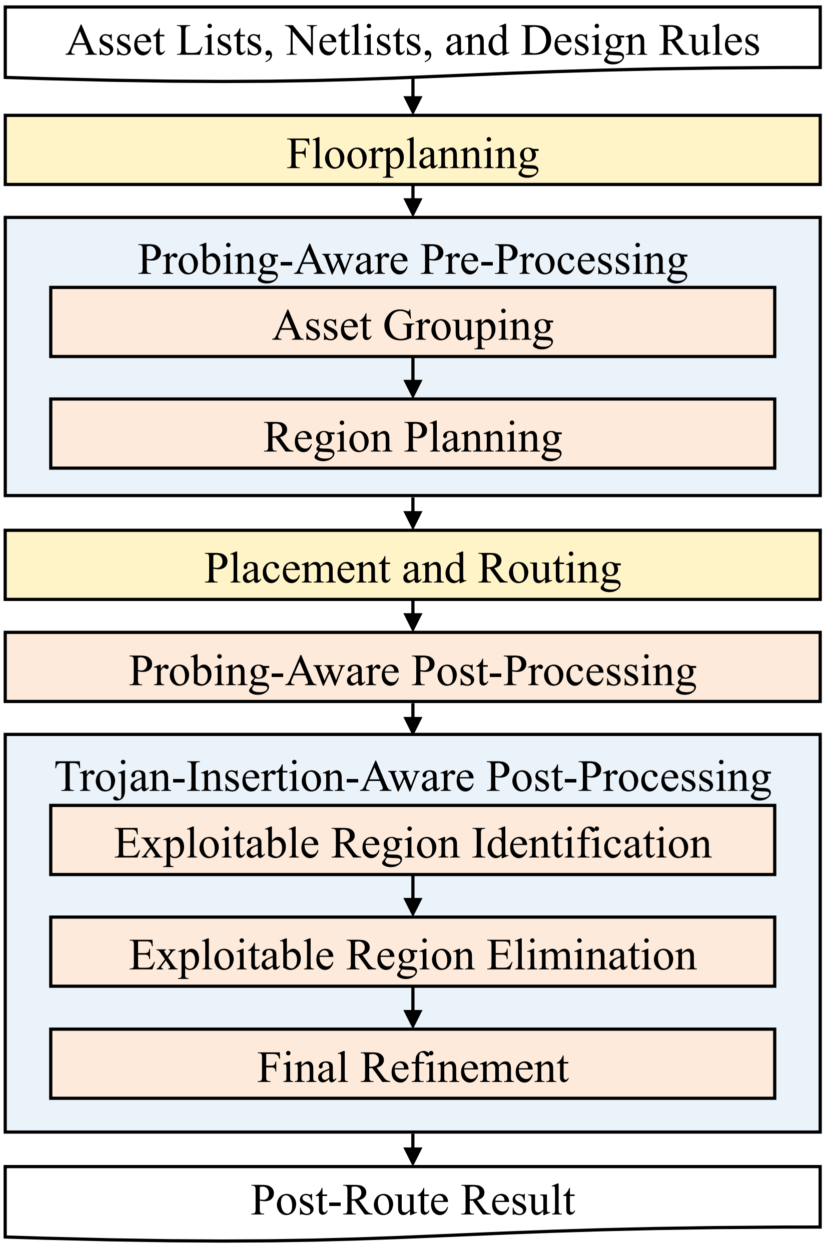
# Our Proposed Framework

In this chapter, we first provide an overview of our framework in Section 3.1. Then, the methods used in each stage are detailed in Sections 3.2–3.6.

## 3.1 Framework Overview

This section provides an overview of our proposed framework, outlining how we tackle the security-aware physical layout problem. Figure 3.1 illustrates our proposed security-aware design framework, which contains five stages: (1) *Floorplanning*, (2) *Probing-Aware Pre-processing*, (3) *Placement and Routing*, (4) *Probing-Aware Post-processing*, and (5) *Trojan-Insertion-Aware Post-Processing*. In floorplanning, we increase design utilization and maintain the overall structure of the power delivery network. Then we construct the placement regions and routing blockages in probing-aware pre-processing, and then perform region-aware placement and regular routing. In probing-aware post-processing, we create a large-scale shielding net to cover the whole assets. Finally, in Trojan-Insertion-Aware Post-Processing, we perform our greedy method to eliminate exploitable regions.

doi:10.6342/NTU202202298



Asset Lists, Netlists, and Design Rules
Floorplanning
Probing-Aware Pre-Processing
Asset Grouping
Region Planning
Placement and Routing
Probing-Aware Post-Processing
Trojan-Insertion-Aware Post-Processing
Exploitable Region Identification
Exploitable Region Elimination
Final Refinement
Post-Route Result

Figure 3.1: Overview of our proposed framework.

## 3.2 Floorplanning

The objective of *Floorplanning* is to determine the appropriate core utilization rate and reconstruct the power delivery network (PDN). First, we extract the PDN information from the original netlist. Then apply a high core utilization rate to decrease exploitable regions preliminarily. Finally, we reconstruct the PDN to meet the constraint mentioned in Section 2.5.

Exploitable regions are one of the primary criteria for a Trojan insertion attack. A naive method to reduce exploitable regions is to apply a high core utilization rate, as shown in Figure 3.2. Increasing the core utilization rate reduces exploitable regions and area costs significantly. However, a high core utilization rate also leads to the difficulty of timing closure. Based on the experiments, we set the target core utilization rate to 90%–95% for each case to prevent timing closure failures.

Most modern EDA tools will perform netlist reconstruction at the placement stage to better performance in timing and area metrics. However, performing netlist reconstruction may lead to the cell or net assets being removed or restructured. This will break the constraint mentioned in Section 1. So at this stage, we apply the "dont_touch" attribute to the cells and nets in the asset list. We do not apply the "dont_touch" attribute to every cell and net in the asset list because too many constraints will make EDA tools spend more time on optimization. We only apply the "dont_touch" attribute to cells that are buffers or inverters, and nets connected to this cell.

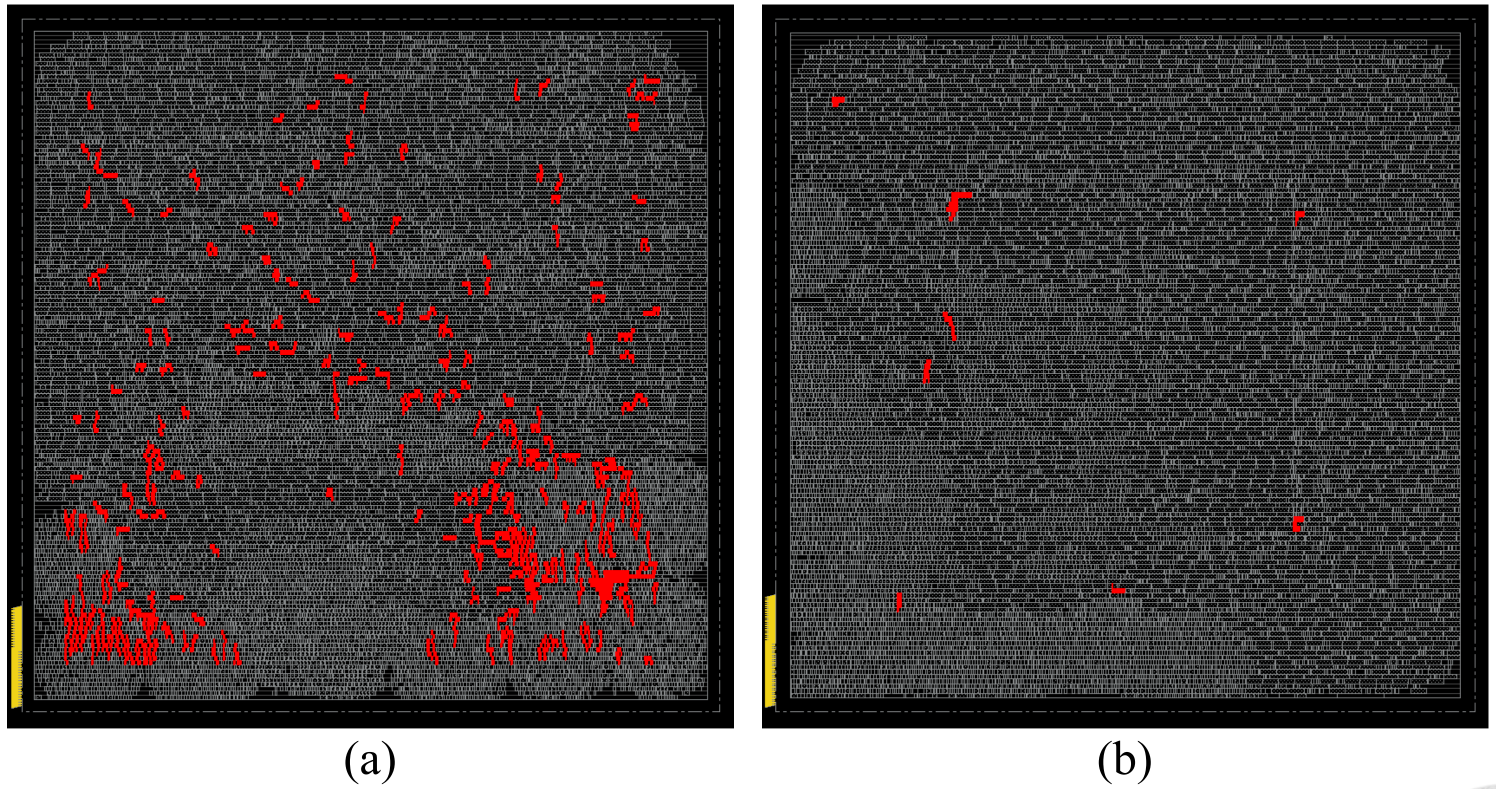


Figure 3.2: A comparison of layouts with different utilization rates. (a) A layout with a 75% utilization rate. (b) A layout with a 94% utilization rate.

## 3.3 Probing-Aware Pre-Processing

*Probing-Aware Pre-Processing* is performed after *Floorplanning*. This stage is to make the placer place the cell assets and cells connected to net assets closer. The pre-processing procedure is divided into three steps: (1) *Asset Grouping* and (2) *Region Planning*.

### 3.3.1 Asset Grouping

Most state-of-the-art placers use routability-driven placement to address the congestion issue during placement. Many previous works proposed various techniques to deal with congestion problems, such as white spacing allocation [5], pin density control [9], and cell inflation [10]. These techniques increase the separation between cells to preserve more routing resources. However, protecting cell and net assets that are spread out all over the layout is complicated and inefficient. Therefore, we group both cell assets and cells connected to net assets and assign a higher weight to net assets to make assets closer and shorter.

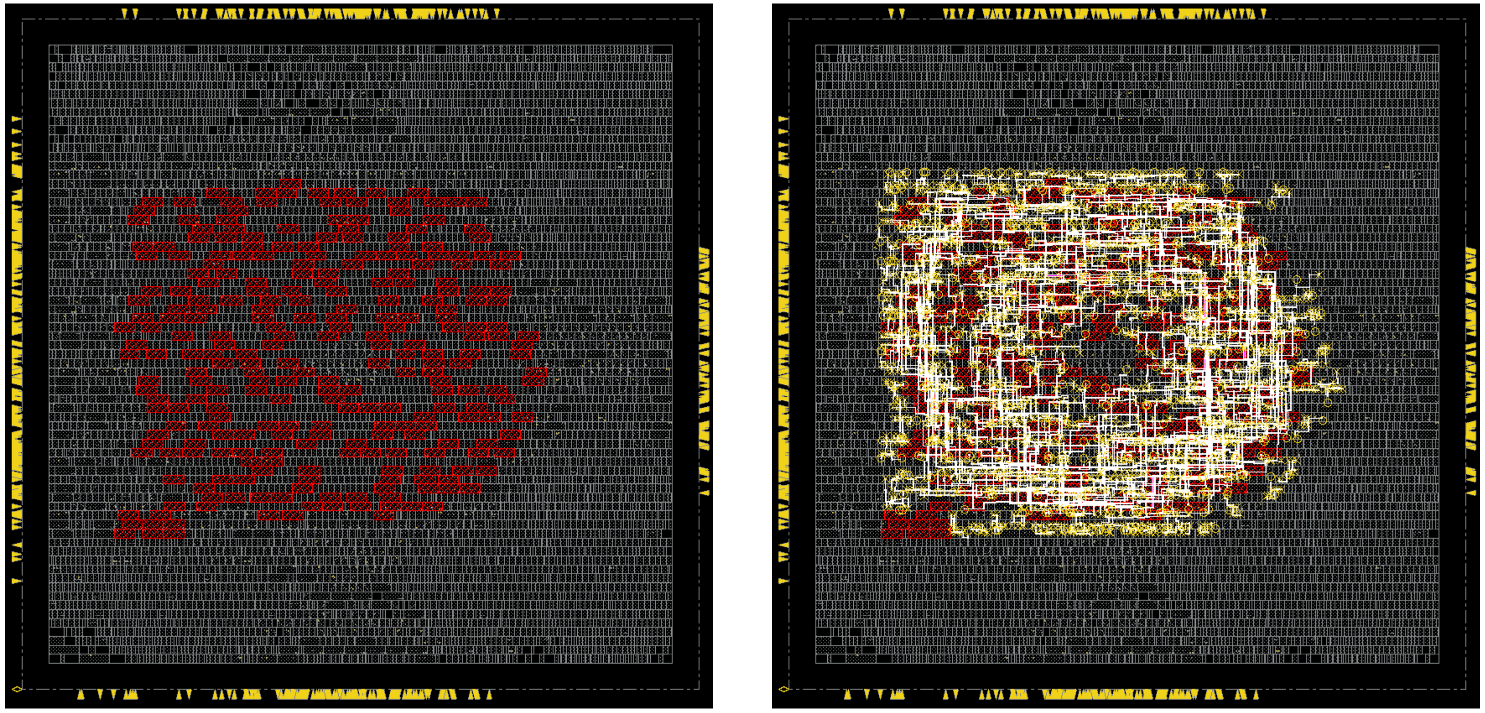
### 3.3.2 Region Planning

Region Planning aims to construct a placement and routing blockage. Algorithm 1 summarizes our planning algorithm, and the algorithm is detailed as follows. First, for each cell in the asset group, we calculate the summation of all cell areas in Lines 1–3. Next, we determine the asset region size based on the area ratio of the asset group area and the core area in Lines 4–8. $\alpha_x$ and $\alpha_y$ are two user-defined parameters to adjust the position of the region of the asset group. Finally, Lines 9–12 determine the position and size of a routing blockage by expanding the asset group region to prevent the regular routing wires from getting too close to the shielding net created later. In our implementation, the parameter $m$ is set to the minimum value that satisfies a *parallel run length spacing* constraint.

**Algorithm 1** Region Planning

```
Input: a asset group $G_a$, margin $m$, $\alpha_x$, $\alpha_y$
Output: a asset group region and a routing blockage region
 1: $assets\_area \leftarrow 0$
 2: for each cell $c_i$ in $G_a$ do
 3:     $assets\_area \mathrel{+}= c_i.area$
 4: $ratio \leftarrow sqrt(assets\_area/core.area)$
 5: $assets\_region.ll.x \leftarrow core.ll.x + \alpha_x * (1 - ratio)/2 * core.width$
 6: $assets\_region.ur.x \leftarrow core.ur.x - \alpha_x * (1 - ratio)/2 * core.width$
 7: $assets\_region.ll.y \leftarrow core.ll.y + \alpha_y * (1 - ratio)/2 * core.height$
 8: $assets\_region.ur.y \leftarrow core.ur.y - \alpha_y * (1 - ratio)/2 * core.height$
 9: $blockage\_region.ll.x \leftarrow assets\_region.ll.x - m$
10: $blockage\_region.ur.x \leftarrow assets\_region.ur.x + m$
11: $blockage\_region.ll.y \leftarrow assets\_region.ll.y - m$
12: $blockage\_region.ur.y \leftarrow assets\_region.ur.y + m$
```

doi:10.6342/NTU202202298



(a) (b)

Figure 3.3: A layout example after performing asset grouping and region planning. (a) Cell assets are marked as red. (b) Net assets are highlighted.

## 3.4 Placement and Routing

After performing probing-aware pre-processing, the blockage information is passed to the placer and router, and region-aware placement and routing is performed. The region-aware placement and routing are performed by a leading commercial tool in our implementation. At this stage, any placer and router that honors region constraints can also be used.

## 3.5 Probing-Aware Post-Processing

This step aims to create a large-scale shielding net to cover all cell and net assets. First, we verify if all the net assets are routed within the routing blockage boundary constructed at the probing-aware pre-processing stage. If there exists any net asset which is routed out of a blockage boundary, these net assets are deleted. Then, we perform rip-up and reroute to ensure that all net assets are routed inside the blockage boundary. Second, we remove the routing blockage and select a high slack net that is not a net asset to create a large-scale shielding net in the top metal layer to cover both cell and net assets. The size of a shielding net is the same as a routing blockage.

## 3.6 Trojan-Insertion-Aware Post-Processing

Even though using a high core utilization rate can effectively reduce the majority of exploitable regions (Section 3.2), a few exploitable regions may still remain. To eliminate remaining exploitable regions, we apply a cell moving-based greedy algorithm. The algorithm contains three steps: (1) *Exploitable Region Identification*, (2) *Exploitable Region Elimination*, and (3) *Final Refinement*.

doi:10.6342/NTU202202298

### 3.6.1 Exploitable Region Identification

**Algorithm 2** Exploitable Region Identification

```
Input: a placed design, a cell assets list $L_c$, exploitable distance $d_e$, threshold $t$
Output: a set of exploitable region list $S_e$
1: Initialize boundary and $S_e$ variables
2: for each cell $c_i$ in $L_c$ do
3:    if $c_i.x < boundary.ll.x$ then
4:       $boundary.ll.x \leftarrow c_i.x$
5:    else if $c_i.x > boundary.ur.x$ then
6:       $boundary.ur.x \leftarrow c_i.x$
7:    if $c_i.y < boundary.ll.y$ then
8:       $boundary.ll.y \leftarrow c_i.y$
9:    else if $c_i.y > boundary.ur.y$ then
10:      $boundary.ur.y \leftarrow c_i.y$
11: $boundary.ll.x -= d_e$; $boundary.ur.x += d_e$
12: $boundary.ll.y -= d_e$; $boundary.ur.y += d_e$
13: for each row $r_i$ in rows intersects the boundary do
14:    for each site $s_i$ in $r_i$ do
15:       if $s_i$ is empty, unmarked, and inside the boundary then
16:          $(num\_site, site\_list) \leftarrow$ perform BFS($s_i$)
17:          if $num\_site >= t$ then
18:             $S_e \leftarrow site\_list$
19: return $S_e$
```

This step aims to identify all exploitable regions to be removed. We first identify the smallest bounding box covering all cell assets. Next, we find the rows that intersect with the bounding box and then iteratively scan an empty site. Finally, we perform breadth-first search (BFS) starting at the empty site to check if the site can form a region with more than the Trojan-placement threshold.

Algorithm 2 summarizes our identification algorithm. We first initialize variables *boundary* that store lower-left and upper-right coordinates, where $S_e$ stores a set of exploitable region lists to be returned by Algorithm 2. Next, for each cell in the cell assets list, find the lower-left and upper-right coordinates of all cell assets in Lines 2–10. Then, we expand the bounding box based on a given exploitable distance $d_e$ in Lines 11–12 because exploitable regions are only evaluated within an exploitable distance related to cell assets. Finally, Lines 13–18 for each row intersects the expanded bounding box, iterates each empty site, and performs BFS to check if the site can form an exploitable region. If so, we add a site list returned by the BFS algorithm to $S_e$ and return it.

### 3.6.2 Exploitable Region Elimination

This stage is to eliminate exploitable regions identified in Section 3.6.1. Due to the effort in Section 3.2, most designs should only remain sporadic exploitable regions. Thus, we propose a fast cell-movement-based method to eliminate the exploitable regions greedily. The objective is to maximize the gain, which is defined as the reduction of the total number of sites in exploitable regions caused by cell movement.

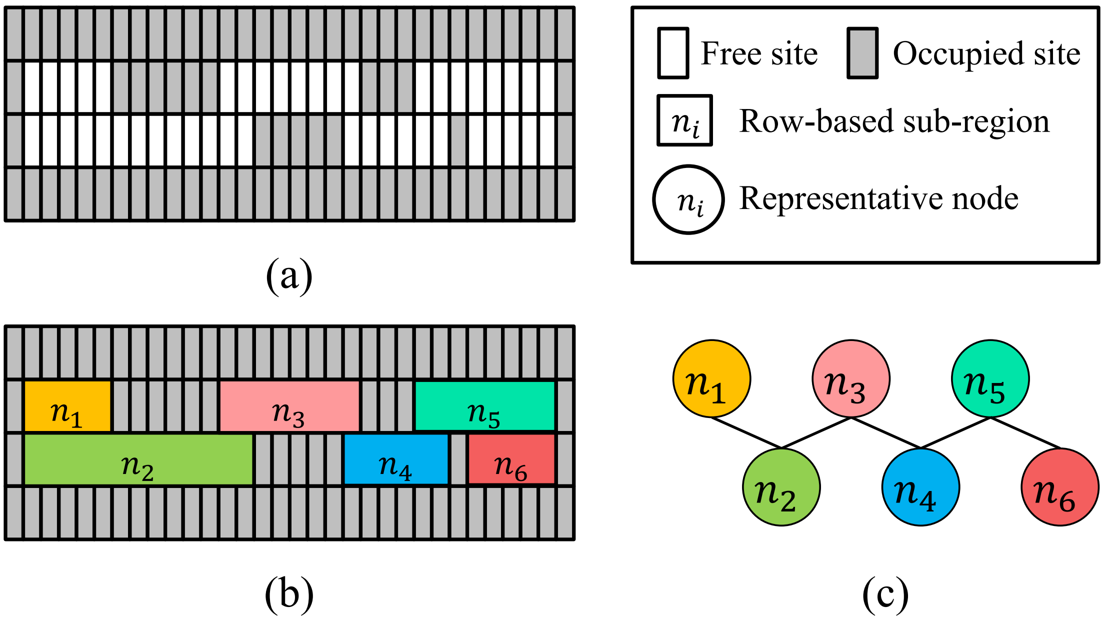


Figure 3.4: (a) An example of an exploitable region. (b) Row-based sub-regions of the exploitable region. (c) The graph transformed from (b).

First, a row-based graph $G$ is constructed, as shown in Figure 3.4(c). We partition an exploitable region into sub-regions, as shown in Figure 3.4(b) and then transform each sub-region into a node with weight equal to the number of sites in the sub-region. Also, edges are constructed to represent the connection of consecutive sub-regions in adjacent rows.

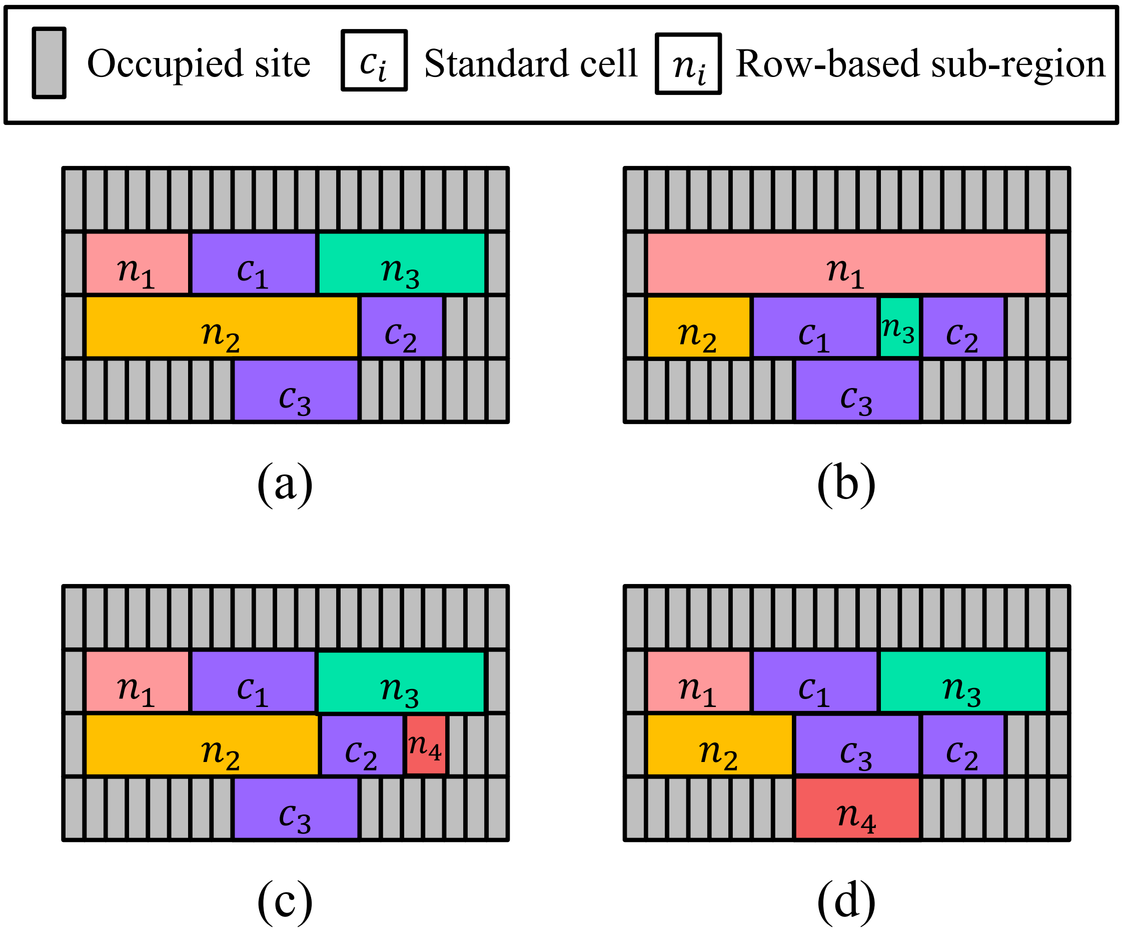


Figure 3.5: An example of cell movement process to illustrate how to determine which sub-region should be cut. (a) The original placement. (b) $c_1$ is moved to cut $n_2$. (c) $c_2$ is moved to cut $n_2$. (d) $c_3$ is moved to cut $n_2$.

Next, the exploitable regions are eliminated by separating them into regions with the number of continuous sites less than the Trojan-placement threshold. We initialize the cost with the number of sites in the exploitable region, which is the summation of nodes' weights in $G$. For each node, we compute the gains obtained by moving the unlocked cell around it to the free sites and select the node with the maximal gain. For example, we calculate the gain for node $n_2$ by moving the cell $c_1$, $c_2$, and $c_3$, respectively, from the original placement, as shown in Figure 3.5(a), deriving candidates, as shown in Figures 3.5(b)–(d), and select $n_2$ due to the highest gain of Figure 3.5(d). After that, we determine if the movement operation is legal by checking whether it creates new exploitable regions. The cell will be moved to the designated sites if the movement holds, and $G$ will be updated. Otherwise, we do not perform an actual movement. Then, the cell is locked. The process is repeated until the exploitable regions are removed, or the cells around exploitable regions are all locked.

Our experimental results show that the algorithm can effectively remove most of the exploitable regions. In rare cases, the remaining ones will be tackled in Section 3.6.3.

### 3.6.3 Final Refinement

If exploitable regions still exist after executing *Exploitable Region Elimination*. It implies that standard cells around these exploitable regions are too small or too sparse to divide the exploitable regions through cell movement. Thus, upsizing nearby cells and/or inserting dummy buffers are applied to remove all remaining exploitable regions.

doi:10.6342/NTU202202298

# Chapter 4

# Experimental Results

In this chapter, we show the experimental results of our proposed framework for the security- aware layout design problem. Section 4.1 introduces the experimental setup. Next, Section 4.2 shows the experimental results of the proposed framework compared to the top-3 teams in the 2022 ACM ISPD Security Closure of Physical Layouts Contest.

## 4.1 Experimental Setup

Our framework was implemented in the Tcl script language. We conducted all experiments on an AMD Ryzen 3990X 2.9GHz Linux workstation with 128GB memory. We used *Cadence Innovus 18* for physical design from floorplan to routing. We adopted the benchmark from the 2022 ACM ISPD Security Closure of Physical Layouts Contest, where Table 4.1 gives the benchmark statistics, including the number of cells ("# cells"), the number of cell assets ("# cell assets"), the number of nets ("# nets"), the number of net assets ("# net assets") , utilization rate ("Utilization"), and the number of metal layers ("# Layers"). The benchmarks contain different crypto core designs and microcontrollers with varying ranges of complexity, timing constraints, utilizations, and the number of assets. All designs were synthesized by the *Synopsys Design Compiler* with the *Nangate 45nm Open*

doi:10.6342/NTU202202298

*Cell Library* [1]. Except for the AES series cases, other cases are limited to six metal layers for routing.

To evaluate the proposed framework, we used the evaluation scripts and the cost metric provided by the 2022 ACM ISPD Security Closure of Physical Layouts Contest. The cost metric is defined as follows:

$$overall\ cost = \frac{ti + fsp\_fi}{2} \times des, \tag{4.1}$$

where $ti$, which stands for the cost of Trojan insertion attack, is the average of placement sites and routing resources of exploitable regions, $fsp\_fi$, which denotes the cost of frontside probing and fault injection attacks, is the average of exposed area of standard cell and net assets, and $des$, which represents the design cost, is the mean of power, performance, area, and the number of design rule checking (DRC) violations. All metrics are normalized to their respective nominal baselines. This means that an improvement to a layout in some metrics will get a value between 0 (maximum improvement) and 1 (minimum improvement).

doi:10.6342/NTU202202298

| | # cells | # cell assets | # nets | # net assets | Utilization | # Layers |
|---|---|---|---|---|---|---|
| AES_1 | 16509 | 336 (2.04%) | 19541 | 468 (2.39%) | 75.1% | 10 |
| AES_2 | 16509 | 336 (2.04%) | 19541 | 468 (2.39%) | 75.1% | 10 |
| AES_3 | 15836 | 322 (2.03%) | 18868 | 461 (2.44%) | 94.8% | 10 |
| Camellia | 6710 | 265 (3.94%) | 7094 | 384 (5.41%) | 51.1% | 6 |
| CAST | 12682 | 1886 (14.87%) | 13050 | 1919 (14.70%) | 51.3% | 6 |
| MISTY | 9517 | 256 (2.68%) | 9895 | 14 (0.14%) | 51.6% | 6 |
| PRESENT | 868 | 164 (18.89%) | 1030 | 145 (14.07%) | 50.6% | 6 |
| SEED | 12682 | 4810 (37.92%) | 13055 | 4938 (37.83%) | 51.37% | 6 |
| TDEA | 2269 | 168 (7.40%) | 2592 | 168 (6.48%) | 81.12% | 6 |
| SPARX | 128 | 8146 (1.57%) | 10786 | 2176 (20.17%) | 50.99% | 6 |
| openMSP430_1 | 4690 | 1541 (32.85%) | 5310 | 1393 (26.23%) | 50.46% | 6 |
| openMSP430_2 | 5921 | 1839 (31.05%) | 6307 | 1717 (27.22%) | 80.09% | 6 |

Table 4.1: Benchmark Statistics.

## 4.2 Results and Comparisons

In this section, we first compare our proposed framework to the top-3 teams in the 2022 ACM ISPD Security Closure of Physical Layouts Contest in Section 4.2.1. Then, we give a design cost analysis in Section 4.2.2.

### 4.2.1 Comparison with the top-3 winner teams

Table 4.2 shows the experimental results. Our framework fully protects the physical design layout and achieves the best scores in terms of the security cost. Furthermore, we still obtained the best design score compared with all the participating teams. Specifically, compared with the top-3 teams, we achieved a better average score with 0.5%, 8.1%, and 12.8% smaller design costs, respectively. In particular, we obtained the smallest design costs for seven out of 12 benchmark circuits (marked in bold).

In our observation, the robustness of our framework against the attacks is contributed by our two-stage security-aware processing strategy, which provides a global view in the early stage before placement and the local view in the latter stage after routing. For Trojan insertion attack, we primarily tackle the problem by increasing the utilization rate in the floorplanning stage, which reduces the possible exploitable regions and provides sufficient resources for Trojan-Insertion-Aware Post-Processing. For frontside probing and fault injection attacks, we conduct the probing-aware region planning before placement, which reserves the routing space for a large-scale shielding net and assures that the cell and net assets are placed and routed within the shielding region.

Furthermore, our better design scores result from our proper trade-off between security and conventional design costs. That is, we do not over-emphasize

doi:10.6342/NTU202202298

security and sacrifice the design cost. In all security-aware processing, we are also dedicated to minimizing the overhead of design costs. For example, in the region planning stage of probing-aware pre-processing, we determine a suitable region considering power, performance, and area. Also, in Trojan-Insertion-Aware Post-Processing, we conduct the cell-movement method prior to cell-resizing and/or buffer-insertion because the overhead of the former method is smaller than the latter ones.

doi:10.6342/NTU202202298

### 4.2.2 Design Cost Analysis

Table 4.3 gives the design cost statistics of layouts, including the design cost ("des"), the total power ("des_p_total"), the area ("des_area"), the number of DRC violations ("des_DRC"), and setup time total negative slack (TNS) ("des_perf"). The "des_perf" is defined as follows:

- $des\_perf = setup\_TNS(submission)/setup\_TNS(baseline)$
  $if\ setup\_TNS(submission) < 0\ and\ setup\_TNS(baseline) < 0$
- $des\_perf = abs(setup\_TNS(submission))$
  $if\ setup\_TNS(submission) < 0\ and\ setup\_TNS(baseline) >= 0$
- $des\_perf = 0$
  $if\ setup\_TNS(submission) >= 0\ and\ setup\_TNS(baseline) < 0$

Compared to the baseline, we achieved an improvement of 37.1% on average in area cost. Even though we increased the core utilization rate by up to 90%, all cases still have no DRC violations and timing violations. This shows that our framework is able to give proper guidance so that EDA tools can find better solutions in the optimization process while taking the security metric into account.

doi:10.6342/NTU202202298

| Benchmark | 1st place | | | 2nd place | | | 3rd place | | | Ours | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | des | ti | fsp_fi | des | ti | fsp_fi | des | ti | fsp_fi | des | ti | fsp_fi |
| AES_1 | 0.447645 | 0.000000 | 0.000000 | 0.475469 | 0.000000 | 0.000000 | 0.519014 | 0.000000 | 0.000000 | **0.427583** | 0.000000 | 0.000000 |
| AES_2 | **0.425056** | 0.000000 | 0.000000 | 0.458233 | 0.000000 | 0.000000 | 0.509517 | 0.000000 | 0.000000 | 0.438185 | 0.000000 | 0.000000 |
| AES_3 | **0.473199** | 0.000000 | 0.000000 | 0.498813 | 0.000000 | 0.000000 | 0.541594 | 0.000000 | 0.000000 | 0.485633 | 0.000000 | 0.000000 |
| CAST | 0.412035 | 0.000000 | 0.000000 | 0.409304 | 0.000000 | 0.000000 | 0.439133 | 0.000000 | 0.000000 | **0.396717** | 0.000000 | 0.000000 |
| Camellia | **0.398203** | 0.000000 | 0.000000 | 0.420739 | 0.000000 | 0.000000 | 0.418330 | 0.000000 | 0.000000 | 0.405498 | 0.000000 | 0.000000 |
| MISTY | 0.418306 | 0.000000 | 0.000000 | 0.396844 | 0.000000 | 0.000000 | 0.417127 | 0.000000 | 0.000000 | **0.387017** | 0.000000 | 0.000000 |
| PRESENT | **0.359781** | 0.000000 | 0.000000 | 0.427651 | 0.000000 | 0.000000 | 0.446817 | 0.000000 | 0.000000 | 0.363447 | 0.000000 | 0.000000 |
| SEED | 0.416061 | 0.000000 | 0.000000 | 0.442646 | 0.000000 | 0.000000 | 0.442510 | 0.000000 | 0.000000 | **0.414805** | 0.000000 | 0.000000 |
| SPARX | 0.397067 | 0.000000 | 0.000000 | 0.420406 | 0.000000 | 0.000000 | 0.404258 | 0.000000 | 0.000000 | **0.387243** | 0.000000 | 0.000000 |
| TDEA | **0.459273** | 0.000000 | 0.000000 | 0.526013 | 0.000000 | 0.000000 | 0.524128 | 0.000000 | 0.000000 | 0.484775 | 0.000000 | 0.000000 |
| openMSP430_1 | 0.406426 | 0.000000 | 0.000000 | 0.440711 | 0.000000 | 0.000000 | 0.469361 | 0.000000 | 0.000000 | **0.401875** | 0.000000 | 0.000000 |
| openMSP430_2 | 0.464010 | 0.000000 | 0.000000 | 0.543684 | 0.000000 | 0.000000 | 0.570014 | 0.000000 | 0.000000 | **0.460260** | 0.000000 | 0.000000 |
| average | 0.423089 | - | - | 0.455043 | - | - | 0.475150 | - | - | **0.421087** | - | - |
| ratio | 1.005 | - | - | 1.081 | - | - | 1.128 | - | - | 1.000 | - | - |

Table 4.2: Comparison of solution quality. The smallest design costs are marked in bold.

| **Benchmark** | **des** | **des_p_total** | **des_area** | **des_DRC** | **des_perf** |
|---|---|---|---|---|---|
| AES_1 | 0.427583 | 0.921278 | 0.789055 | 0 | 0 |
| AES_2 | 0.438185 | 0.962943 | 0.789797 | 0 | 0 |
| AES_3 | 0.485633 | 0.953148 | 0.989380 | 0 | 0 |
| CAST | 0.396717 | 1.020214 | 0.566652 | 0 | 0 |
| Camellia | 0.405498 | 1.059360 | 0.562629 | 0 | 0 |
| MISTY | 0.387017 | 0.979187 | 0.568883 | 0 | 0 |
| PRESENT | 0.363447 | 0.899424 | 0.554360 | 0 | 0 |
| SEED | 0.414805 | 1.071469 | 0.587750 | 0 | 0 |
| SPARX | 0.387243 | 0.999080 | 0.549888 | 0 | 0 |
| TDEA | 0.484775 | 1.101294 | 0.825972 | 0 | 0 |
| openMSP430_1 | 0.401875 | 1.039979 | 0.564785 | 0 | 0 |
| openMSP430_2 | 0.46026 | 0.993591 | 0.843006 | 0 | 0 |
| Average | 0.421087 | 1.000081 | 0.682680 | 0 | 0 |

Table 4.3: Design cost statistics of layouts.

doi:10.6342/NTU202202298

# Chapter 5

# Conclusions and Future Work

In this chapter, Section 5.1 give a conclusion of our work. Next, we provide several future works in Section 5.2.

## 5.1 Conclusions

We have proposed a security-aware framework against Trojan insertion, frontside probing attacks, and fault injection attacks while minimizing extra design costs. A large-scale shielding method has been proposed to cover all the exposed cell and net assets. A cell movement method has also been proposed to eliminate the exploitable regions with low overheads. Experimental results have shown that our proposed framework achieves the highest score among all participating teams in the 2022 ACM ISPD Contest.

doi:10.6342/NTU202202298

## 5.2 Future Work

Some future research directions are provided as follows:

### 5.2.1 Consider Different Types of Attacks

In this thesis, we only consider three types of attacks: (1) Trojan insertion, (2) frontside probing attacks, and (3) fault injection attacks. However, there are many other types of attacks, such as side channel attacks, that pose a severe threat to the security of the chip. Side channel attacks are a type of attack that exploits information leaked from the chip. Figure 5.1 shows an overview of side channel attacks. This type of attack can be done through many channels, such as time analysis [14], power consumption [19], and thermal behavior [16]. This type of attack analyzes the current operational behavior of the chip by analyzing the information emitted from the chip. For example, if an attacker uses power consumption analysis to accomplish an attack, we can make the power consumption of most operations on the chip the same to make analysis difficult. Therefore, we believe this is a good research topic for future advanced process design flows.

doi:10.6342/NTU202202298

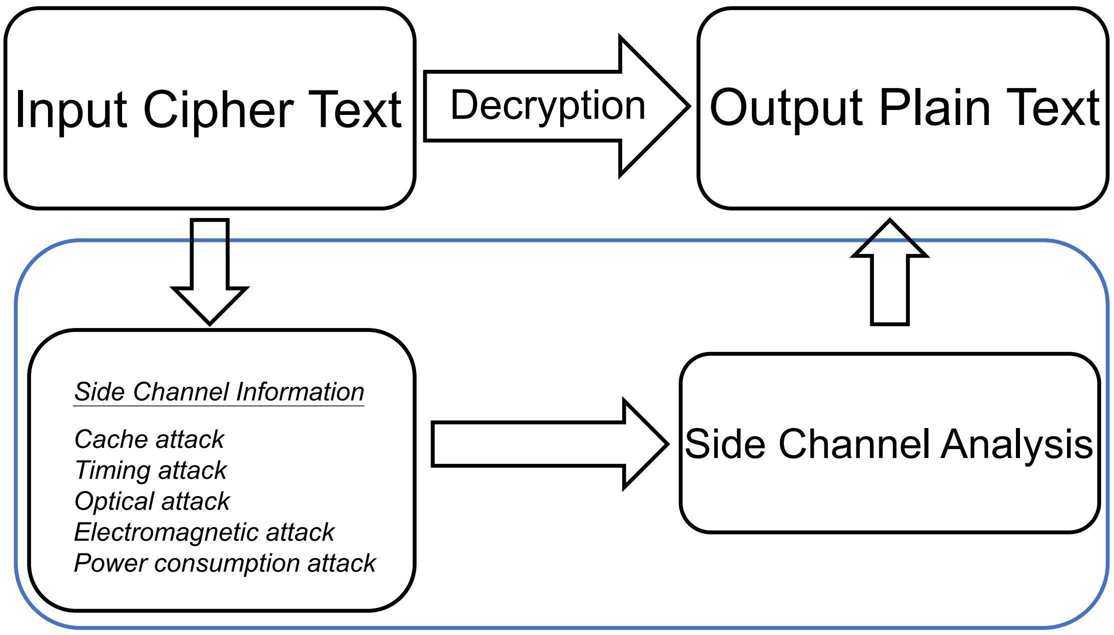


Figure 5.1: A high-level overview of side channel attacks [24].

### 5.2.2 Security-Aware Placement and Routing

In this thesis, our proposed framework is to influence the algorithm of placement and routing in EDA tools by giving commands to the tools. However, this approach is very limited and inflexible in what it can do. It can easily affect the optimization process of the algorithm to the extent that better results are not achieved. Therefore, we believe that it is better to consider the security-related metrics directly in the optimization process of the placement or routing algorithms. For example, in the routing stage, we can route the sensitive nets on the bottom layer and the rest of the nets on the upper layer, then the routing pattern can be similar to the sensitive nets or use a special pattern to route the nets, to block the sensitive cells and net as much as possible. Figure 5.2 shows an example of a special pattern routing.

doi:10.6342/NTU202202298

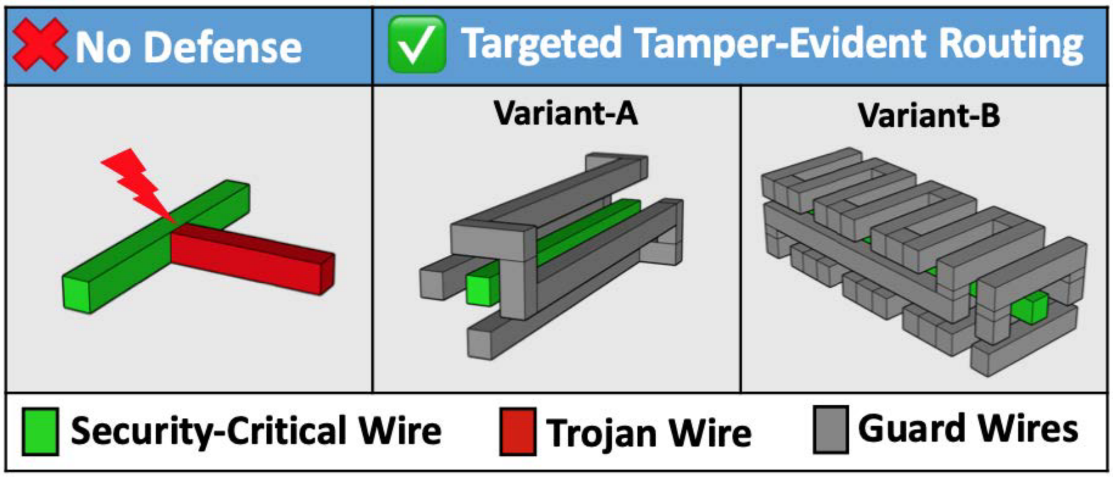


Figure 5.2: An example of a special pattern routing [22]

# Bibliography

[1] Nangate freepdk45 open cell library. [Online]. Available: https://si2.org/open-cell-library/

[2] Security closure of physical layouts. [Online]. Available: https://wp.nyu.edu/ispd_22_contest/

[3] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache, "Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures," *Proceedings of the IEEE*, vol. 100, no. 11, pp. 3056–3076, 2012.

[4] M. Beaumont, B. Hopkins, and T. Newby, "Hardware trojans-prevention, detection, countermeasures (a literature review)," 2011.

[5] L. Chen, X. Min, C.-K. Koh, J. Cong, and P. H. Madden, "Routability-driven placement and white space allocation," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pp. 394–401, 2004.

[6] J.-M. Cioranesco, J.-L. Danger, T. Graba, S. Guilley, Y. Mathieu, D. Naccache, and X. T. Ngo, "Cryptographically secure shields," in *IEEE International Symposium on Hardware-Oriented Security and Trust*, pp. 25–31, 2014.

[7] X. Guo, R. G. Dutta, J. He, M. M. Tehranipoor, and Y. Jin, "QIF-Verilog: Quantitative information-flow based hardware description languages for pre-silicon security assessment," in *IEEE International Symposium on Hardware Oriented Security and Trust*, pp. 91–100, 2019.

doi:10.6342/NTU202202298

[8] N. Homma, Y.-i. Hayashi, N. Miura, D. Fujimoto, D. Tanaka, M. Nagata, and T. Aoki, “EM attack is non-invasive? - design methodology and validity verification of em attack sensor,” in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 1–16, 2014.

[9] M.-K. Hsu, Y.-F. Chen, C.-C. Huang, S. Chou, T.-H. Lin, T.-C. Chen, and Y.-W. Chang, “NTUplace4h: A novel routability-driven placement algorithm for hierarchical mixed-size circuit designs,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 12, pp. 1914–1927, 2014.

[10] C.-C. Huang, H.-Y. Lee, B.-Q. Lin, S.-W. Yang, C.-H. Chang, S.-T. Chen, Y.-W. Chang, T.-C. Chen, and I. Bustany, “NTUplace4dr: A detailed-routing-driven placer for mixed-size circuit designs with technology and region constraints,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 3, pp. 669–681, 2018.

[11] Y. Jin and Y. Makris, “Hardware trojan detection using path delay fingerprint,” in *IEEE International Workshop on Hardware-Oriented Security and Trust*, pp. 51–57, 2008.

[12] M. Khairallah, R. Sadhukhan, R. Samanta, J. Breier, S. Bhasin, R. S. Chakraborty, A. Chattopadhyay, and D. Mukhopadhyay, “DFARPA: Differential fault attack resistant physical design automation,” in *Proceedings of ACM/IEEE Design, Automation and Test in Europe*, pp. 1171–1174, 2018.

[13] J. Knechtel, J. Gopinath, J. Bhandari, M. Ashraf, H. Amrouch, S. Borkar, S.-K. Lim, O. Sinanoglu, and R. Karri, “Security closure of physical layouts iccad

doi:10.6342/NTU202202298

special session paper," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–9, 2021.

[14] P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre attacks: Exploiting speculative execution," in *IEEE Symposium on Security and Privacy*, pp. 1–19, 2019.

[15] E. Love, Y. Jin, and Y. Makris, "Proof-carrying hardware intellectual property: A pathway to trusted module acquisition," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, 2012.

[16] R. J. Masti, D. Rai, A. Ranganathan, C. Müller, L. Thiele, and S. Capkun, "Thermal covert channels on multi-core platforms," in *Proceedings of USENIX Conference on Security Symposium*, pp. 865—-880, 2015.

[17] M. Nabeel, M. Ashraf, S. Patnaik, V. Soteriou, O. Sinanoglu, and J. Knechtel, "2.5D root of trust: Secure system-level integration of untrusted chiplets," *IEEE Transactions on Computers*, vol. 69, no. 11, 2020.

[18] X. T. Ngo, J.-L. Danger, S. Guilley, T. Graba, Y. Mathieu, Z. Najm, and S. Bhasin, "Cryptographically secure shield for security ips protection," *IEEE Transactions on Computers*, vol. 66, no. 2, 2017.

[19] M. Randolph and W. Diehl, "Power side-channel attack analysis: A review of 20 years of study for the layman," *Cryptography*, vol. 4, no. 2, 2020.

[20] C. Shepherd, K. Markantonakis, N. van Heijningen, D. Aboulkassimi, C. Gaine, T. Heckmann, and D. Naccache, "Physical fault injection and side-channel attacks on mobile devices: A comprehensive analysis," *Computers & Security*, vol. 111, p. 102471, 2021.

doi:10.6342/NTU202202298

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

[22] T. Trippel, K. G. Shin, K. B. Bush, and M. Hicks, "T-TER: Defeating A2 trojans with targeted tamper-evident routing," 2019.

[23] T. Trippel, K. G. Shin, K. B. Bush, and M. Hicks, "ICAS: an extensible framework for estimating the susceptibility of ic layouts to additive trojans," in *IEEE Symposium on Security and Privacy*, pp. 1742–1759, 2020.

[24] R. Vanathi and S. Chokkalingam, "Side channel attacks in IaaS and its defense mechanisms," *International Journal of Engineering and Advanced Technology*, vol. 8, 2019.

[25] H. Wang, D. Forte, M. M. Tehranipoor, and Q. Shi, "Probing attacks on integrated circuits: Challenges and research opportunities," *IEEE Design & Test*, vol. 34, no. 5, pp. 63–71, 2017.

[26] H. Wang, Q. Shi, A. Nahiyan, D. Forte, and M. M. Tehranipoor, "A physical design flow against front-side probing attacks by internal shielding," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2152–2165, 2019.

[27] X. Wang, M. Tehranipoor, and J. Plusquellic, "Detecting malicious inclusions in secure hardware: Challenges and solutions," in *IEEE International Workshop on Hardware-Oriented Security and Trust*, pp. 15–19, 2008.

[28] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware trojans: Lessons learned after one decade of research," *ACM Transactions on Design Automation of Electronic Systems*, vol. 22, no. 1, pp. 1–23, 2016.

doi:10.6342/NTU202202298

[29] K. Xiao, D. Forte, and M. Tehranipoor, “A novel built-in self-authentication technique to prevent inserting hardware trojans,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 12, 2014.

doi:10.6342/NTU202202298

# Publication List

[1] Y. Chou, J.-W. Hsu, Y.-W. Chang, and T.-C. Chen, "VLSI Structure-aware Placement for Convolutional Neural Network Accelerator Units," in *Proceedings of ACM/IEEE Design Automation Conference*, December 2021.

doi:10.6342/NTU202202298