Golden-Chip Free Side Channel Delay Analysis Test for Hardware Trojan and Recycled IC Detection

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Ashkan Vakil
Master of Science
University of Bridgeport, 2015

Director: Dr. Avesta Sasan, Professor
Department of Electrical and Computer Engineering Department

Spring Semester 2021
George Mason University
Fairfax, VA

Dedication

I dedicate this dissertation to my family, who support me during the challenges throughout my journey; to the challenge, that confronting it leads to improvement; and lastly to improvement itself, that without it I could not realize the preciousness of time, which is the most valuable.

Acknowledgments

I would like to express the deepest appreciation to my advisor, Dr. Sasan, who has continued supporting me during this chapter of my life.

Table of Contents

List of Tables

List of Figures

Abstract

GOLDEN-CHIP FREE SIDE CHANNEL DELAY ANALYSIS TEST FOR HARDWARE TROJAN AND RECYCLED IC DETECTION

Ashkan Vakil, PhD

George Mason University, 2021

Dissertation Director: Dr. Avesta Sasan

The distributed manufacturing supply chain of Integrated Circuits (IC) introduces many vulnerabilities during IC's life cycle. An adversary in an untrusted foundry can exploit these weaknesses to design malicious hardware attacks that target the integrity, reliability, and trustworthiness of fabricated ICs.

This work introduces a set of physical-aware and learning-assisted modeling techniques, followed by test methodologies, for Hardware security in the post-fabrication stage. The proposed detection approach targets to identify Hardware-Trojan infected chips and recycled-ICs. Unlike the prior art, this flow does not require a Golden fabricated chip as a fingerprint to compare the side-channel signals. Instead, by modeling the voltage drop and voltage noise pre-fabrication and training a Neural Network post-fabrication, our proposed technique can improve the timing model collected during timing closure and produces a Neural Assisted Golden Timing Model (NGTM) for side-channel delay-signal analysis.

The Neural Network acts as a process tracking watchdog for correlating the static timing data (produced at design time) to the delay information obtained from a clock frequency sweeping test. Proposed detection flow enables Hardware Trojan detection close to 90%, and 100% recycled-IC detection in the simulated scenarios.

Chapter 1: Introduction

In the past decade, to reduce the fabrication cost and for economic feasibility, the manufacturing supply chain of Integrated Circuits (IC) has adopted a globally distributed model [2]. The use of untrusted entities in this global supply chain has raised pressing concerns about the security of fabricated ICs, including the possibility of IP piracy/theft, Trojan insertion, overbuilding, and counterfeiting [3]. Researches have constantly provided solutions for each threat, i.e. using encryption and activation mechanisms to lock the circuit before authorized application [4–10], although, at the same time, the attack models have evolved into more comprehensive and dormant models. Hardware (HW) Trojan insertion and Counterfeiting are the threats that we target here, and propose modeling and testing methodologies to detect them.

1.1 Hardware Trojan

One of these security threats is the adversarial infestation of fabricated ICs with a HW Trojan. A Trojan can be broadly defined as a malicious modification to a circuit to control, modify, disable, or monitor its logic. The spectrum of harm caused by HW Trojans is broad. It can range from passive Trojans for activity monitoring or stealing information to weaponized Trojans that could cause catastrophic consequences in the critical military, space, or medical applications [11]. Thereby, detecting HW Trojans is highly crucial, and it has become a significant concern for governments and industries.

One solution for detecting HW Trojan is through destructive Reverse Engineering (RE) schemes to check and ensure that the manufactured chips' logical structure and functional integrity is untouched. Relying on RE to produce a golden model from fabricated ICs has severe limitations. The destructive process of de-layering, combined with advanced image

processing techniques could be used for the generation of a netlist, but not a golden model containing all process information (such as doping levels and the extent of parametric variation in that process) as such information can not be extracted using imaging techniques. Besides, IC reverse engineering requires significant investment, is extremely challenging in advanced geometries, and is quite time and resource consuming. One may argue that a Trojan-induced logic-change can be detected during Manufacturing test process. However, HW Trojans are stealthy in nature, and are designed such that they are rarely activated. This makes detecting the Trojans during manufacturing testing highly difficult if not impossible.

Conventional manufacturing VLSI test and verification methodologies fall short in detecting HW Trojans due to the different and un-modeled nature of these malicious alterations. This has led many researchers to investigate solutions for detection of HW Trojans through statistical analysis of side-channel information collected from ICs, including side-channel power analysis [1, 12–16], power supply transient signal analysis [17, 18], regional supply currents analysis [19], temperature analysis [20], wireless transmission power analysis [21], and side-channel delay analysis [22–28].

The problem with many of the previous HW Trojan detection solutions is the need for some sort of a golden model from which the parametric signature of the fabricated ICs are collected and used to define a decision boundary (power, delay, temperature, etc) for separating the Trojan-infested ICs. However, building a golden IC is extremely difficult or even impossible: In many cases, especially in advanced technology node, the choice of the foundry is limited to one or a very few, none of which may be trusted. Even if a trusted foundry exists, fabrication of a small volume of ICs for obtaining a golden IC is usually cost prohibitive[15]. Moreover, the process used in each foundry is quite different and a golden IC that is fabricated in one foundry can not be used for assessing an IC fabricated in another foundry.

1.2 Recycled IC

A counterfeit IC can be an unauthorized copy of the exact IC, a slightly modified version, a recycled IC, or a defective one [29]. Counterfeit ICs have high impact on the IC manufacturing business model, as they have lower performance and more failure over time, when deployed. Recycled ICs, ICs that have been already used but are pretended to be new, have contributed to more than 80% of the counterfeit ICs in recent years[30]; posing around $169 billion revenue loss to the global electronics supply chain[31]. Based on U.S. Chamber of Commerce reports, counterfeit ICs have even found their way into military supply chain [32]. A recycled chip exhibits a lower performance and a higher failure rate over time, since its embedded transistors have already been aged, i.e., their characteristics have been derated due to the chip usage. This increases the probability of chip failure sooner than expected; shortening the chip lifespan. The short lifetime and low performance of the recycled ICs not only affect the end-users but also puts a significant financial burden on the industry and government sectors. Therefore, a concrete solution to detect recycled ICs is highly required.

Among the various designs' robustness concerns affecting CMOS devices, aging effects have been receiving a lot of attention. In practice, aging mechanisms degrade the reliability and performance of CMOS devices over their lifetime. Due to aging, the electrical behavior of transistors deviates from their original intended behavior, resulting in performance degradation and the ultimate chip failure [33, 34]. Among aging mechanisms, the effect of Bias Temperature Instability (BTI) and Hot Carrier Injection (HCI) are more dominant than other aging mechanisms [35]. In this work, we focus on these two aging mechanisms when detecting the recycled chips.

Guardbanding is the current industrial practice to cope with transistor aging and voltage droops. It entails slowing down the clock frequency (i.e., adding timing margin during design) based on the worst degradation the transistors might experience during their lifetime [36]. The guardbands ensure that the chip functionality is intact for an average period

of 5 to 7 years. However, inserting wide guardbands degrades performance and increases energy consumption. Hence, chip design companies usually have small guardbands, typically 5-10% [37]. Fig. 1.1 shows an overview of the guardband assignment for each chip during the manufacturing. In this figure, the delay of the critical path is C_0 when the chip is new. As the chip is used, its critical path delay gradually increases due to aging; reaching T_0 after a period of t = Y years. Accordingly, for the chip to be usable during its expected lifetime (Y Year), it needs to be clocked at a frequency no higher than $F_0 = 1/T_0$. Thus, the designers add a guardband $G = T_0 - C_0$ to prevent any aging-induced chip malfunction during its expected lifespan (i..e, Y years in Fig. 1.1). However, if the chip is a recycled one, the remaining life expectancy is less than Y, i.e., it can experience a malfunction much sooner than the expected lifetime (Y).

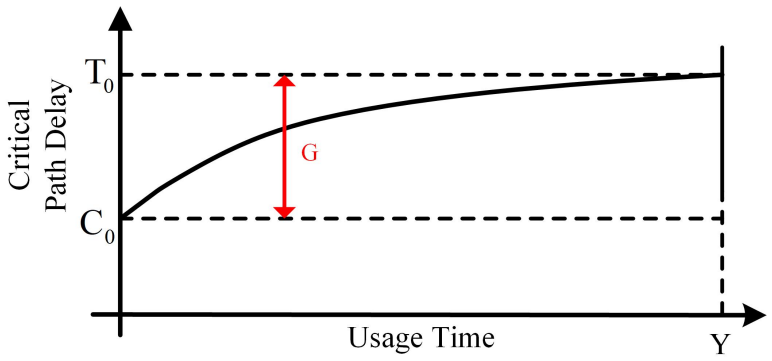


Figure 1.1: Impact of aging on critical path delay.

Device workload as well as environmental conditions, including temperature and voltage source, all affect the device aging rate. In fact, each logical cell residing in a chip is aged differently based on the duration of logical '1' and '0' values applied to its input pins, or the toggling count that its transistors observe.

1.3 Challenges and Proposed Solution

The application of side-channel information of a trusted model in security test has been a research direction for many years. The trusted model can either be obtained using a fabricated trusted chip (golden chip), or a trusted time-closed design. As mentioned earlier, having a fabricated golden chip is costly and sometimes not possible for higher technology nodes. On the other hand, the variation that happens during fabrication process has high impact in more advance technology node. Knowing this, in case of using a trusted time-closed model as the golden model, the side-channel information should account for this variation; meaning the proposed detection solution should be able to differentiate the variation-induced change in side-channel signals from what has been added due to a security threat. Otherwise, ignoring the impact of variation would result in false positives or false negatives, which in turn, for instance in case of counterfeit test, it might results in disposing a new IC or deploying a recycled one, respectively.

In side-channel with delay signal analysis, using side-channel information of a trusted netlist faces serious challenges: The change in the delay of timing paths could also be the result of (systematic and random) Process Variation (PV) and/or process drift. PV is the *unintended* variation in the physical and electrical property (strength) of transistor devices due to the physical limitation of manufacturing devices in building perfect transistors. Process drift is the *intended* change in the process over time, made by the fabrication lab to improve the process. Although such improvement guarantees a working transistor made using Spice models generated for an older version of the process, the strength and characteristic of devices (and in the result their speed) would change over time. Therefore, by having access to the original GDSII and timing model (generated for the original Spice Model), a tester can not determine if the change in side-channel information is due to PV and/or process drift or if it is due to a hardware attack.

In our proposed flow, we do not assume the existence of a golden fabricated IC. Instead, we develop and train a Neural Assisted Golden Timing Model (NGTM) post-fabrication

combined with a voltage variation modeling during the Static Timing Analysis (STA) to make the NGTM fingerprint. This NGTM model is then used in a security test to be compared with the side-channel signals collected from the suspicious circuit. The security test, then assigns a tag to each IC under the test and indicate if the IC is clean. The threat that our security test is targeting specifies how to choose instances for training the model, and defines the structure of our security test.

When the security test targets to detect Trojan threat, the side-channel signals are tested per timing-path and for all paths in the chip under the test. In case of Recycled IC detection, the security test aims to differentiate between aged and new devices by assessing the impact of aging on the delay of carefully selected timing paths in a chip under test. Different timing paths, for having different topologies, for being made of different types of devices, and for experiencing different switching activities would age differently. Hence, in an aged IC, one could expect to see a disparity in the aging-induced increase in the delay across timing-paths. Therefore, the security test focuses on analyzing the average of aging-induced-delay of groups of timing-paths, rather than being per timing-path.

1.4 Motivation

This work is motivated by two previous papers: The side-channel power analysis in [15] and side-channel delay analysis in [22], a short description of which is given next:

The side-channel statistical power analysis solution for Trojan detection in [15] proposed that the trusted region for the operation of a Trojan free IC can be learned using a combination of a trusted simulation model, measurements from the carefully engineered and distributed Process Control Monitor (PCM) structures, and advanced statistical tail modeling techniques. The mentioned work, however, relies on side-channel power analysis for the detection of hardware Trojan. For observing a meaningful change in leakage or dynamic power, the size of HW Trojan has to be large. Hence, this technique falls short of detecting Hardware Trojans implemented using a small number of gates. This is when our proposed solution can detect even a single added logic gate in a tested timing path.

Besides, [15] relies on the usage of PCMs (with a defined structure that is repeated and distributed over the IC) for extracting the process parameters. However, the number and accuracy of PCMs are limited. Although PCM can roughly track the process corner from chip to chip and could be used for the rough calibration of timing and spice models, they fall short of accurately characterizing the behavior of different gates and metal layers. This is when in our proposed solution, every timing path could be used as a PCM for training the neural assisted timing augmentation engine, and therefore the impact of different timing path topology, different gate types/sizes, and the change in the capacitive or resistive load of different metal layers are taken into account.

The side channel delay analysis solution in [22] uses Clock Frequency Sweeping Test (CFST) to detect the hardware Trojan. However, it relies on the existence of a Golden IC for delay comparison. Our proposed side-channel Trojan detection scheme is inspired by this work (and used CFST for the generation of label data points for each feature set), however, our proposed mechanism does not need a Golden IC.

Chapter 2: Background

2.1 Hardware Trojan

In practice, a HW Trojan can be inserted at any stage of the design flow[38–42]. Upon activation of the Trojan, the Trojan delivers its payload which can result in a denial of service in the whole or part of the circuit, corruption of the circuit's functionality, an alteration in the characteristic of the circuit such as aging factors, or leaking secret information [41, 42].

Countermeasures against HW Trojans can be categorized into the design-for-security, run-time monitoring, and detection solutions [42]. The design-for-security approaches opt to reduce the chances of Trojan insertion (e.g., through hardware obfuscation),or increase the chance of detection. However, they can neither guarantee a Trojan free IC nor detect them. The run-time techniques monitor the functionality of the IC (usually through snapshots of its operation) at run-time [43], and compare it against known behavior signatures. However, if the Trojan impact does not persist, it does not create the expected signature, or affects the IC's behavior in a way that is not modeled (in the monitoring solution), the monitoring schemes will fail to detect the Trojan. On the other hand, detection approaches aim to directly or indirectly detect the presence of HW Trojans. Detection solutions could be destructive or non-destructive [42]. The destructive solutions, that could provide an ultimate proof for Trojan's presence in the selected IC, require full reverse engineering of the IC.

The non-destructive detection approaches can detect the Trojans by either activating them or via side-channel signal analysis [41,42,44]. The former relies on finding a set of input patterns that trigger the possible Trojan such that the Trojan results in a noticeable impact (e.g., change in expected output). On the other hand, the side-channel based detection methods attempt to identify the Trojan presence through side-channel information obtained

from an IC, e.g., power consumption [1,13,14,45], electromagnetic emanations [46], or path delays [22–24].

Detecting Trojans by activating them during manufacturing test is significantly challenging. In principal, Trojans are designed to be activated through a rare sequence or combination of events, only known to the adversary [22, 42]. Testing an IC exhaustively using all sets of possible patterns is not practically feasible [22]. Note that not all HW Trojans alter the functionality. For example, a HW Trojan may be designed to leak secret information (with antenna or noise); making such Trojan immune to activation-based solutions as the functional impact of such HW Trojan is not observable.

Trojan activation solutions' limited coverage has encouraged the research to focus on side-channel analysis-based detection techniques. One widely studied Trojan detection direction is through side-channel power analysis [1,47–50] that focuses on power consumed by The Trojan Circuit (TC). However, for side-channel power analysis, TC should be partially or fully activated. Therefore it is better suited for Trojans, trigger of which is connected to shorter timing paths with a higher degree of controllability. At the same time, the power signature of the TC should be significant enough to stand out (make a noticeable change in the power consumption of the IC) as the demanded current of an IC can be monitored with limited precision (through package balls or, at best case, through power delivery networks pads). Hence, the observed current signature is the accumulation of the transistors' current needed for the normal function of the IC and those added for implementation of TC. Therefore, the size of a TC should be large enough to be observable using such techniques.

The delay side-channel test, on the other hand, focuses on the change of the delay and measures path delays to detect a Trojan [12]. The delay analysis proposed in [23] monitors the critical timing-paths to detect Trojans. However, it fails to consider the near-critical or shorter timing paths. The authors of [24] insert shadow registers to measure the delay of each timing-path. However, this results in a large area overhead. Finally, the solution suggested by [22] uses Clock Frequency Sweeping Test (CFST) to detect Trojans. However, it relies on the existence of a Golden IC for delay comparison. This work inspires our

proposed side-channel Trojan detection scheme; however, our proposed solution relaxes the need for the presence of a Golden (trusted) fabricated IC.

2.2 Recycled IC

Recycled IC detection methods can be classified into several categories. The first category deals with conventional test methods that perform physical (e.g. detecting repackaged ICs using 3-Dimensional imaging technologies) and electrical (studying the ICs' parameters such as threshold voltage, path delays, etc) tests [51] [52]. Such tests are conducted in specific labs following several testing standards such as AS6171, AS5553, and CCAP-101 [53]. These methods are costly, time-consuming, and have a low detection rate.

The second category of Recycled IC detection schemes, i.e., statistical approaches, mainly deploy machine-learning models to differentiate new and aged chips from each other. For instance, in [54], Ke Huang et al. used a Support Vector Machine (SVM) based technique to classify the chips into recycled versus new using parametric measurements collected from a set of brand new chips including Iddq, F_{max}, V_{min}, etc. These measurements are usually collected in the test facilities to verify the correct functionality of the chips before shipping them to customers. On the other hand, in [55], the authors detect recycled ICs based on the aging rates in similar components that may have resided in the target chip. The assumption is that if the device is aged, similar components in the device (e.g., different full adders in an N-bit adder module) may age differently as they are exposed to different input patterns during run time. In this method, the correlation of dynamic supply current ($IDDT$) between similar logic blocks is calculated, based on which, the IC is reported as new or recycled. The main drawback of the prior art statistical solutions is their reliance on the availability of a golden chip.

The third group, the so-called DFAC (Design For Anti Counterfeiting) strategies, detect the recycled ICs via on-chip sensors[52]. The on-chip sensor-based approaches try to compare the frequency of an embedded element, e.g., a Ring Oscillator (RO) with a reference

point to identify the recycled ICs. For instance, Guin et al. [56] proposed to insert two ring oscillators inside the chip, one is not used frequently while the other is always on. Comparing the frequency of these oscillators reveals if the chip is recycled. As this is sensitive to PV, its detection accuracy is low. To resolve this issue, [31] tried to mitigate the impact of PV by replacing the reference RO with a Non Volatile Memory (NVM) that stores the frequency of the RO when it is new. The stored value is compared with the frequency of the RO when the device is checked regarding its freshness. Any discrepancy demonstrates that the device is not new. To prevent tampering such NVM, the authors proposed to use digital signature verification (e.g., chip unique ID). This method suffers from high power consumption related to its always-on RO.

Aging mechanisms including Bias Temperature Instability (BTI), Hot-Carrier Injection (HCI), Time-Dependent Dielectric Breakdown (TDDB), and Electromigration (EM) result in performance degradation and eventual failure of digital circuits over time [57]. Among all, BTI and HCI are the two leading factors in the performance degradation of digital circuits [35]. Both mechanisms increase the switching delay of transistors, leading to an increase in path delays.

BTI Aging: BTI aging includes Negative Bias Temperature Instability (NBTI) and Positive Bias Temperature Instability (PBTI). NBTI affects a PMOS transistor when a negative voltage is applied to its gate. A PMOS transistor experiences two phases of NBTI depending on its operating condition. The first phase, the so-called *stress phase*, occurs when the transistor is on ($V_{gs} < V_t$). In this case, positive interface traps are generated at the Si-SiO_2 interface which leads to an increase of the threshold voltage of the transistor. The second phase, denoted as the *recovery phase*, occurs when the transistor is off ($V_{gs} > V_t$). The threshold voltage drift that occurred during the stress phase will partially recover in the recovery phase.

Threshold voltage drifts of a PMOS transistor under stress depend on the physical parameters of the transistor, supply voltage, temperature, and stress time [58]. The last three parameters (known as external parameters) are used as acceleration factors of the

aging process. Figure 2.1 shows the threshold voltage drift of a PMOS transistor that is continuously under stress for 6 months and a transistor that alternates between stress and recovery phases every other month. As shown, the NBTI effect is high in the first couple of months but the threshold voltage tends to saturate for long stress times. It is noteworthy to mention that PBTI affects the NMOS transistors in a similar fashion that NBTI affects the PMOS transistors. Accordingly, for the sake of space, we do not discuss PBTI separately.

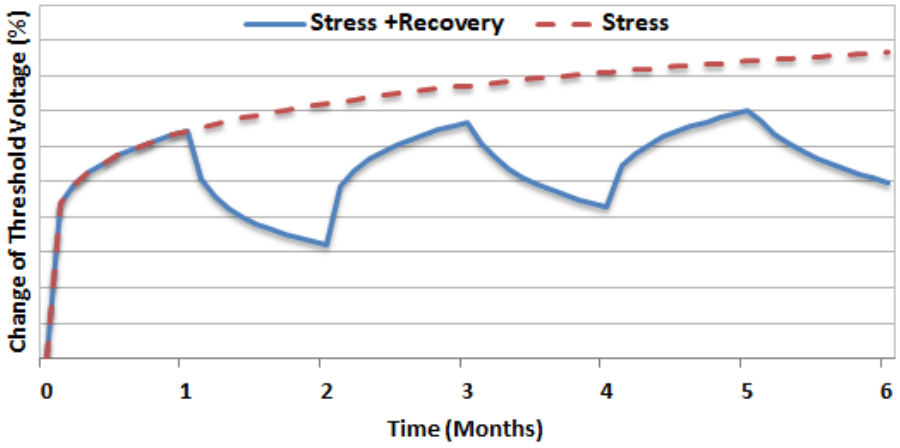


Figure 2.1: Threshold-voltage shift of a PMOS transistor under the NBTI effect. Values on the Y-axis are not shown to make the graph generic for different technologies.

HCI Aging: HCI occurs when hot carriers are injected into the gate dielectric during transistor switching and remain there. HCI is a function of switching activity and degrades the circuit by shifting the threshold voltage and the drain current of transistors under stress. HCI mainly affects NMOS transistors.

HCI-induced threshold voltage drift is highly sensitive to the number of transitions occurring in the gate input of the transistor under stress. In practice, HCI has a sublinear dependency on the clock frequency, usage time, and activity factor of the transistor under stress, where activity factor represents the ratio of the cycles the transistor is switching and the total number of cycles the device is utilized. HCI effect is exacerbated as the operating temperature increases [35].

Chapter 3: Threat Model and Challenges

3.1 Threat Model

The adversary in this research is an untrusted foundry with access to GDSII (Graphic Database System format).

3.1.1 Trojan Threat Model

The goal of the adversary is to insert a Trojan that is triggered based on a combination, or a sequence of rare events. A Trojan, As illustrated in figure 3.1, consists of 1) Trojan's Trigger inputs (TT), 2) Trojan's Triggering (which could be sequential or combinational) Circuit (TTC), and 3) Trojan Payload (TP). Upon activation, the TP alters the circuit functionality. We assume that no Golden IC exists, and the Trojan is inserted in all fabricated ICs.

3.1.2 Recycled IC Threat Model

Similar to the Trojan threat model, we assume that the IC designer is trustworthy while the supply chain is untrusted. In particular, we assume that an adversarial supplier can

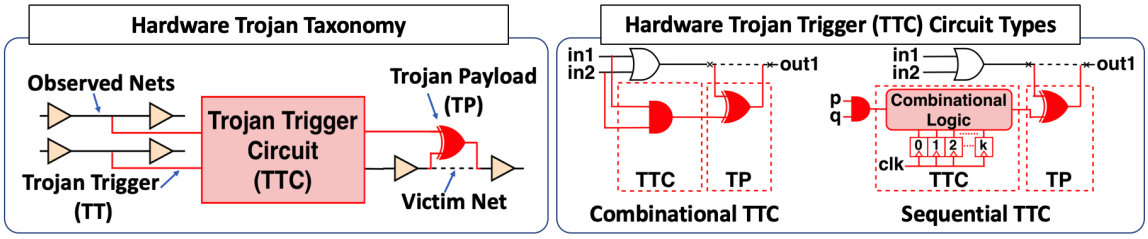


Figure 3.1: (left): Trojan taxonomy, (right): Trojan trigger circuit types

potentially provide the system integrator with a recycled IC, the usage of which (in a critical application) may result in catastrophic consequences (due to the effect of aging on the chip's reliability). We assume that the IC/system designer has access to the chip netlist and its GDSII file. However, she does not have a golden chip to use it as a basis to determine if the chip-under-test is recycled or new.

3.2 Side Channel Detection Challenge: Variability

The TT of an HW Trojan poses an additional capacitive load on its driving cell, resulting in a slower rise and fall, while its TP adds a gate delay to its victim timing path. In a perfect world, A Trojan can be detected by tracking and analyzing the changes in the delay of timing-paths compared to that predicted by STA.

The impact of aging on the threshold voltage of transistor devices within an IC causes shift in the timing behavior of leaf cells. This shift in timing increases the delay of timing-paths. The added delay is not uniformly distributed for all timing paths, as some paths go through more switching activities. One can utilize the different aging behavior of timing-paths and indicates if the chip is brand new with observing the delay-differences of timing-paths in STA and the suspicious chip under the test.

The challenge for this solution is that STA suggested delay information can be significantly different from delay information that is collected at the test time. This is due to several factors most notable of which are: 1- voltage noise, 2- Process Variation (random and systematic variations), and 3- process drift. Thus, it is not determined if the difference between STA and test results is due to a security threat, or these variations.

Voltage Noise: In an ASIC chip, the current flow to and return from transistors via the Power Delivery Network (PDN), which consist of a sequence of resistive, inductive and capacitive elements. A flow of current through a resistive element manifests a voltage (IR) drop that is proportional to the current flow (I) and element's resistance (R). Furthermore, the current flow is orchestrated by the die switching activity that changes per clock cycle

[59]; Hence, due to inductive nature of PDN, transistors also experience an inductive voltage drop which is proportional to the PDN's inductance profile (L) and the rate of change in the current, denoted by $d(i)/d(t)$, which exacerbates at scaled geometries with increased current demand and higher frequencies [60]. In addition, there are both intentional and device/metal topology related decoupling capacitance (DECAP), that decouple the power and ground lines. This results in the PDN to act as an RLC network. In the result, the voltage that a transistor experience is below the voltage supplied from the voltage regulator and also changes dynamically from cycle to cycle causing variation in the delay of timing paths [61]. The cycle to cycle voltage variation, which in physical design flow is denoted by voltage noise, causes clock jitter [62] leading to uncertainty in clock arrival time to the clock pin of registers.

During STA the IR drop and voltage noise are modeled by (1) specifying a rail voltage value below supplied voltage to account for IR drop, and (2) using register-endpoint uncertainty to guard against voltage-variation-induced clock network jitter [62]. The chosen values for the rail-voltage and uncertainty should be pessimistic to capture the worst-case (to prevent setup/hold timing failure). However, the majority of timing-paths experience smaller IR-drop and voltage noise [37]. This poses a security threat; the pessimistic margins build large unused timing slack into the majority of timing-paths, which is not visible to the physical designer and test engineer. The unused timing slacks can be used by an adversary in an untrusted foundry to design a Trojan and hide its delay impact.

Random Process Variation: The random process variation refers to the variations in the physical and electrical properties of transistors due to the physical limitations faced during the fabrication process [63]. The random process variation impacts the delay and drive strength of fabricated transistors and makes threat detection more difficult as the test engineer needs to differentiate between the delays imposed by random process variation and the timing impact of either an HW Trojan or aging-induced delay. Figure 3.2 illustrates the effect of the random process variation on the slack of timing paths.

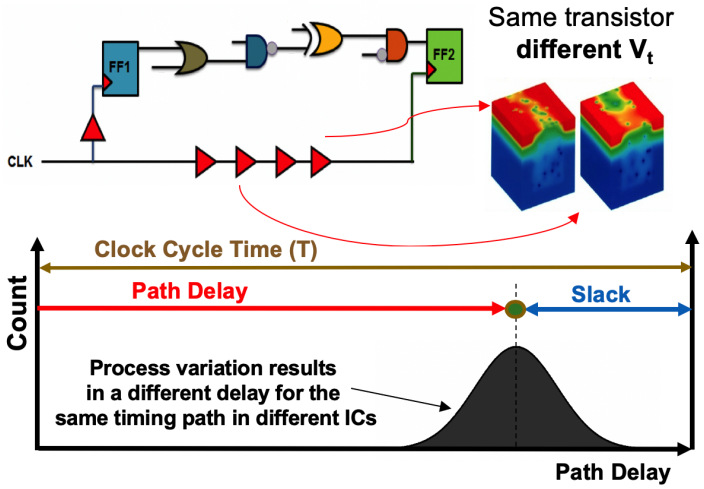


Figure 3.2: The impact of random process variation on the delay of a timing-path when sampled across multiple dies (after fabrication).

Systematic Process Variation: Systematic Process Variation is the result of imperfection in one or several process steps, as a result of which, a systematic shift occurs in the behavior of transistors or wires. For example, the systematic shift may speed up all NMOS transistors, increase the capacitance of a given metal layer, or reduce PMOS transistors' strength. Unlike random process variation (mitigation of which is disclosed when we describe our IC classification methodology), the systematic (inter-die) process variation affects all devices similarly. Therefore, systematic process variation behaves similarly to process drift, with the difference that process drift is the intended consequence of improving the fabrication process. On the other hand, the *systematic process variation is an unintended consequence of imperfection in one or several processing steps.* For example, if during the Chemical Mechanical Polishing step, the height of a specific metal layer, e.g., M4, was less or more than the process defined height, the expected resistance, and capacitance of all M4 net segments systematically shifts. In practice, the systematic process drift can be treated similarly to process drift.

Process Drift: The SPICE model for the fabrication process in a new technology node is released soon after the process is stabled and is used to characterize the standard cell

libraries deployed in a physical design house. The SPICE model and standard cell libraries are padded with carefully crafted margins to guarantee a high yield. Furthermore, the foundry keeps improving the process over time to improve yield and reduce cost and may update the process by deploying newer and more capable stepping devices. Hence, the fabrication process and the released SPICE model drift apart over time. The improvement in the process builds large unused slacks in a fabricated IC that is designed using the older SPICE model. This practice poses a security problem as these unused and hidden timing slacks (to the test engineer) can be used by an adversary in the untrusted foundry to design stealthy HW Trojan(s), or hide the aging-induced delay of recycled ICs. Figure 3.3 illustrates the impact of the Process Drift on the slack of timing paths.

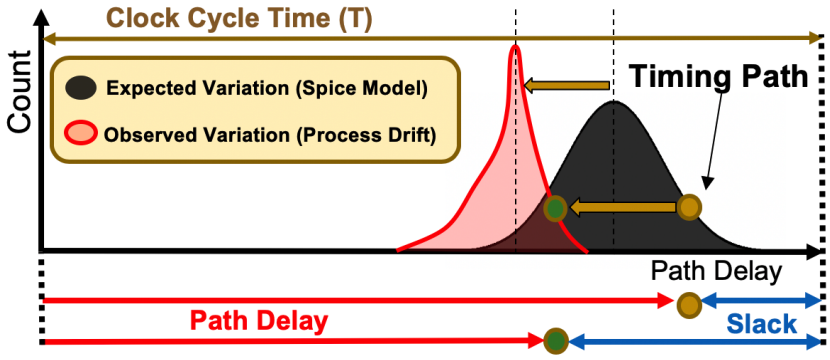


Figure 3.3: Improvement in the process over time non-linearly changes the delay of different timing-paths (process drift). The process drift affects each timing-path differently.

Chapter 4: Proposed Detection Solutions

Proposed detection technique integrates multiple variation modeling and mitigation techniques into a side-channel delay analysis solution for the purpose of HW Trojan testing and recycled IC detection. Using our proposed model, we characterize and mitigate the impact of voltage noise, process variation, and process drift to improve the correlation between the adjusted timing model and the fabricated ICs' timing behavior. This resulted timing model is then used in a classification test to identify the security status of each chip under the test. We first describe how each of these variation sources is modeled and mitigated, and then explain how each mitigation technique is integrated into the proposed scheme to improve the chances of detection.

4.1 Definitions and Model Parameters

Before describing our solution, we elaborate on the model parameters used:

Clock Frequency Sweeping Test (CFST): An existing delay testing solution in which delay of different timing paths is examined while increasing the clock frequency [64]. The target is to find the start to fail frequency for different timing paths. The test accuracy is limited by the tester frequency step size and maximum achievable frequency. The delay reported for each timing path may be affected by both process variation and process drift.

Age Distinguished Paths (ADP): Depending on the circuit topology and workload, some of the timing paths in a circuit age more, and some age less than others. Hence, we can distinguish between two sets of timing paths: 1) Most aging Affected Paths (MAP) and 2) Least aging Affected Paths (LAP).

For simplicity, lets first assume that there is no process drift (but there exist process variation), the step size of the tester equipment is sufficiently small, Static Timing Analysis

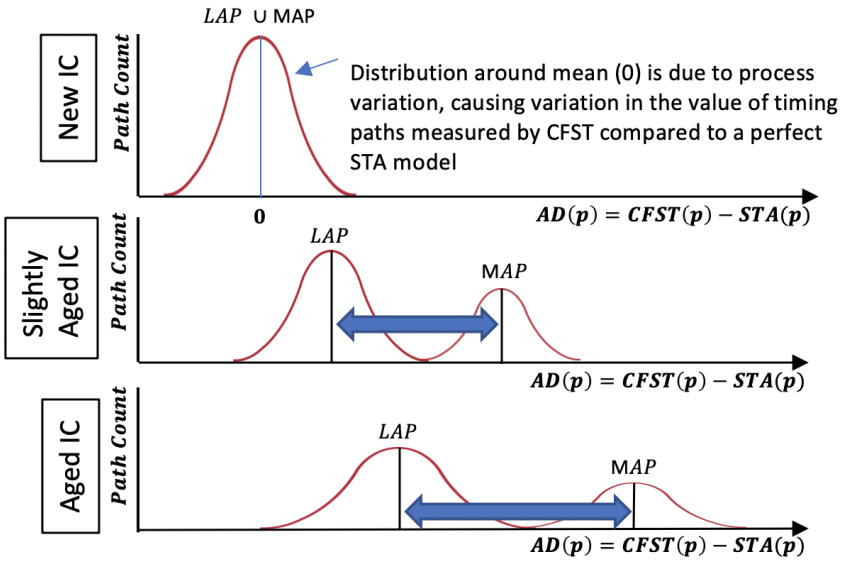


Figure 4.1: In a new device, one cannot distinguish between MAP and LAP timing paths as no aging occurred. Computing the AD for timing paths gives us a zero-mean distribution. As the IC ages, the delay of all timing paths increases, however, the delay-increase is more significant in the MAP set of timing paths. Therefore, the normal distribution of AD morphs into a bimodal distribution as the IC ages. Identification of MAP and LAP sets of timing paths allows us to compute the mean for each set. The shift in the mean is an indication of the extent of aging. In this figure, it is assumed that there is no process drift (but there exist process variation), the step size of the tester is very small, STA is perfect, and CFST reported delay for a timing path at age zero (fresh IC) matches STA. We will update these assumptions to realistic ones when discussing our proposed solution.

is perfect, and CFST reported delay for a timing path at age zero (new IC) matches that of the STA within the boundary of process variation. Let's denote the STA-reported delay of path p with $STA(p)$, and the delay reported by CFST by $CFST(p)$. We define added delay AD for path p as $AD(p) = CFST(p) - STA(p)$. Given a set of timing paths, if we compute AD for each path, we will see a zero-mean normal distribution of ADs if the IC is not aged (in an ideal world). This is illustrated in figure 4.1(top). As the IC ages, the MAP and LAP timing paths would age at different rates (figure 4.1(middle)). Therefore, the normal distribution (observed at age zero) will morph into a bimodal distribution, where the difference in the mean of two clusters increases over time, highlighting the separation between MAP and LAP group. This is the basis for our aged-IC detection.

However, in a real-world, we have to deal with process drift, reduce the impact of process

Table 4.1: Description for each of 48 features, extracted from each timing-path for building the NN training set. (LP: Launch portion of timing-path, CP: Capture portion of timing-path, DP: Data portion of timing-path, M: Metal Layer, x: drive strength of the gate)

Total of 48 Features, 3 Feature Extracted from each timing-path		
Setup Time	Path delay reported in STA	Sum of fanout over cells in DP
45 Feature Extracted, 15 from each sub-path (CP, LP and DP)		
number of gates	subpath Delay	# cells of x0 strength
# cells of x1 strength	# cells of x2 strength	# cells of x4 strength
# cells of x8 strength	# cells of x16 strength	# cells of x32 strength
Total Length of M1	Total Length of M2	Total Length of M3
Total Length of M4	Total Length of M5	Total Length of M6

variation, identify MAP and LAP groups ahead of time, deal with the inaccuracy of tester, account for inaccuracy of the STA and the fact that it does not match the CFST test result. In the next section, we describe our proposed model that deals with each of these phenomenons, for building a reliable aging detection solution.

4.2 Modeling and Tracking the Process Drift

Process drift results in a non-uniform shift in the delay of different timing-paths. To model the timing impact of process drift, we design and train a Neural Network (NN) to act as a process tracking watchdog (NN-Watchdog). This NN-Watchdog is used to predict the difference between the slack reported by STA at design time and that sampled from fabricated IC at test time. To train the NN-Watchdog, we need a labeled data-set. Each data point in our data-set is a collection of 48 input features and a label value. The input features, detail of which is in table 4.1, are extracted from physical design EDA and the STA engine. The label for each data point is the difference between the slack reported by STA (at design time), and that obtained by CFST [22] (at test time).

To assess the effectiveness of NN-Watchdog (and for lack of access to fabricated ICs), we modeled the process drift by extracting the shift in delay values from SPICE simulations performed using a skewed SPICE model. For this purpose, we first extracted the SPICE

model for each timing-path in the input training. Then, to mimic a systematic process drift, the SPICE model was skewed such that the NMOS and PMOS transistors were ∼X% faster, and the Metal capacitance for Metal layers 1 to 7 was derated by Y%. Selection of X and Y gives us a consistently faster or slower process model. For example, the selection of $(X, Y) = (5, 5), (0, 0), (-5, -5)$ produces Fast, Typical, and Slow process models in our simulations.

The resulting dataset was then used to train and evaluate the NN-Watchdog. The threat, that our proposed security test is targeting, defines the way we split the dataset for each of the training and evaluation steps. In case of Trojan detection, the resulting database was then divided into 1) training-set for training the NN (60% of timing-paths), 2) verification-set used for assessing the trained model accuracy while training (20% of timing-paths), and 3) test-set used for reporting the results (20% of timing-paths). In case of recycled IC detection, ADP should set the rules to split dataset for training and test purposes: the training is done using the MAP portion of the dataset, then the NN-watchdog is tested on the LAP portion of the dataset. The reason behind this dataset-split is to bar the NN-Watchdog to learn the impact of aging on MAP and LAP paths. To explain more, let's assume a scenario in which the device has been aged: in this case, both MAP and LAP paths are aged, but MAP paths are experiencing more deviation in delay of those paths. Having the NN-Watchdog trained on only MAP, the model assumes the excessive aging-induced delay is similar for both MAP and LAP paths, although in practice, the LAP paths are aged less. The outcome is having more error in predicting the delay of LAP paths.

In this research, we have evaluated 3 different models to predict the process-induced change in the timing path delays. Details of each model is given next:

(1) Linear Regression (Ridge Regression) Model (baseline): Ridge Regression [65] is a regularized linear regression model and it is useful for modeling and tracking multicollinearity phenomena.

(2) Multi-Layer Perceptron Regression: Multi-Layer Perceptron (MLP), is a non-linear neural network composed of an input layer, one or more hidden layers, and an output

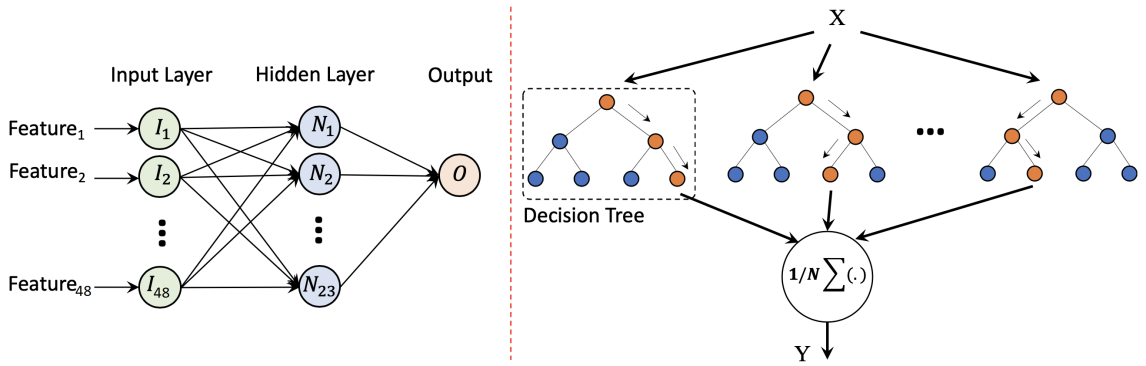


Figure 4.2: Abstract view of a fully-connected NN (left) and a Random forest (right) as two base models to form test-time process watchdog.

layer (figure 4.2-left). Details and setup of MLP regressor used in this work is summarized in table 4.2.

(3) Stacking Regression Model: The structure of Stacking Regression model [66], which is also known as stacked generalization [67], is depicted in Figure 4.3. The Stacking Regression is an ensemble learning technique in which different estimators are arranged into two layers to form a regressor with lower variance in comparison to each (member) regressor. More precisely, at a two-layer stacked regressor (Figure 4.3-top), we used regressors XGB[68], Enet[69], Lasso[70], Ridge[65], MLP[71] and RandomForest[72] for our first layer regression. The predictions of these regressors, $\hat{y}^{\eta_1}$ to $\hat{y}^{\eta_6}$, are stacked together and fed to the second layer of regressor(s). In general, the second layer may also consist of multiple regressors. The overall prediction $\hat{y}^{fin}$ is obtained by averaging the results of the second layer regressors. In our model, we have only deployed a single Lasso [70] regressor in the second layer, as including additional regressor result in only negligible improvement in the model's prediction performance at the cost of increased complexity.

Among the regressors used in two-layer stacked regressors, Enet, Lasso and Ridge are linear. In these models, the difference in the performance stems from their associated regularization penalty. More precisely, Equation 4.1 shows the optimization problem formulated for Enet. Lasso and Ridge are special cases of Enet in which Lasso only considers

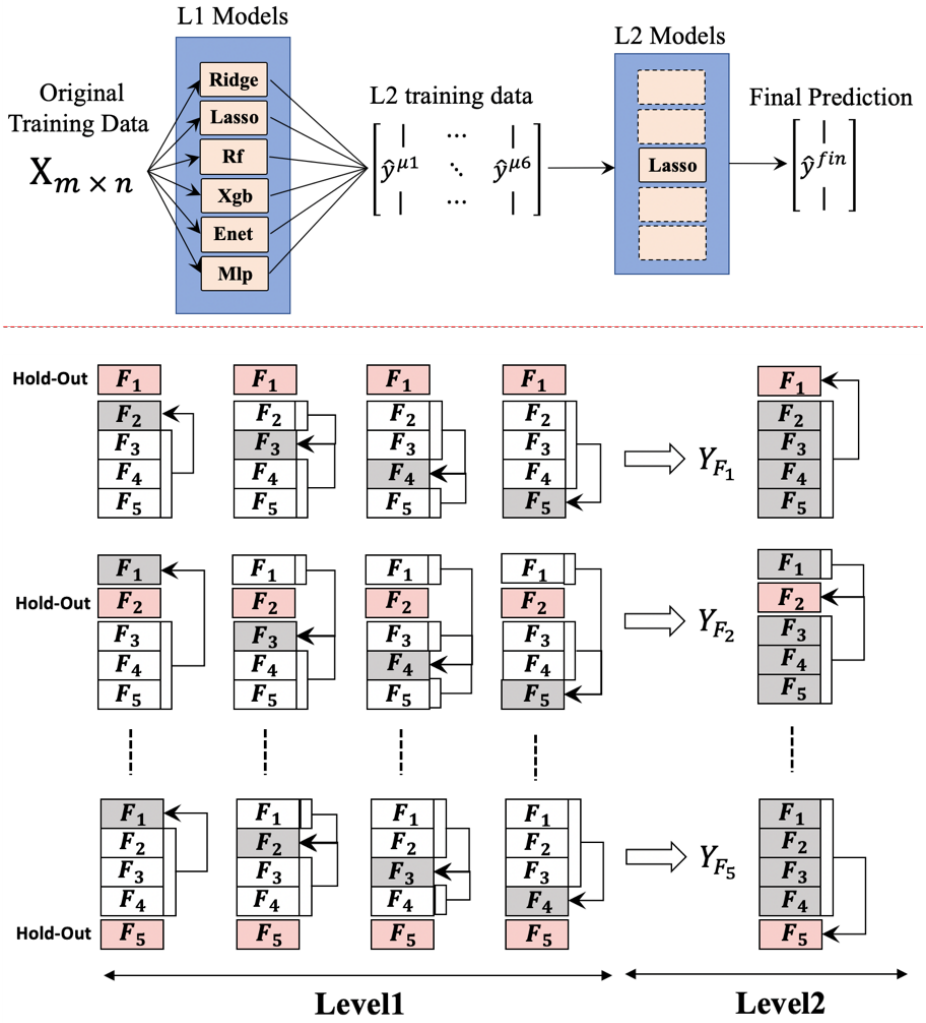


Figure 4.3: Top figure shows a two-layer stacked regressor. Bottom figure shows the cross-validation method used for obtaining hyper parameters at a two layer stacked regressor.

L_1 norm of parameters ($\|W\| = \sum_{j=1}^{N} |W_j|$) while Ridge considers L_2 norm of parameters ($\|W\|_2 = \sum_{j=1}^{N} |W_j|^2$) for regularization.

$$\hat{W} = \underset{W}{argmin}(\|y - XW\|^2 + \lambda_2 \|W\|^2 + \lambda_1 \|W\|) \tag{4.1}$$

Random Forest and XGB can be considered as ensembles of decision-trees. The main difference between these two categories is the way that decision trees are combined. In

Table 4.2: hyper-parameters of regressor models used in this table.

Model	**Hyper-Parameters**
Ridge	alpha=1, max_iter=5000
Lasso	alpha=0.001, max_iter=5000
RF	n_estimators=1024, bootstrap=True, min_leaf=1, min_split=2
Xgb	n_estimators=1024, learning_rate=0.05
Enet	alpha=0.001, max_iter=1000
MLP	in_layer=42, hidden_layer=23, out_layer=1, activation='tanh', optimizer='adam', learning_rate='adaptive', start_lr='0.1'

Random Forest, also known as a bagging-based algorithm, a subset of features is randomly selected to form a forest of decision trees, see figure 4.2-right. Each of these trees is trained independently, and the final regression model is determined by averaging the result of each decision tree. In XGB, also known as a boosting-based algorithm, decision trees depend on each other, and through cascading, the error of previous trees is minimized (Boosting). Details and setup of each regressor used in the stacked model (which is used in this research) are summarized in table 4.2.

Figure 4.3-bottom shows the cross-validation technique for defining the hyper-parameters of each one of the used regressors. Cross-validation consists of four steps: 1) Randomly partitioning the training set into k equal sets, also known as K-fold. 2) Holding out one of the training sets, highlighted with red, from the (k-1)-remaining folds, and training on the (k-2)-fold. The left-out fold, highlighted with gray, is used for validation. This procedure continues for (k-1) times, which results in a stack of prediction of (k-1)-folds, which is stored in Y_{F_i}. 3) Training the layer two regressors based on the obtained dataset, Y_{F_i}, and evaluating the level-2 regressors based on the holdout set, F_i. 4) Selecting the hyper-parameters that result in a lower average loss. Once the hyper-parameters are corrected, both layers are trained on the whole training set, without k-folding, and the final results are reported by evaluating the trained stacked model on the test set.

4.3 Modeling and Mitigating process variation

We divide the process variation into two categories: 1) *Random* Class that includes the independent intra-die process variation, and 2) *systematic* class including all forms of inter-die and correlated intra-die variation.

4.3.1 Systematic Process Variation

We perform speed binning on fabricated ICs and divided them into different speed bins (Fast, Normal, and Slow), arguing that ICs in the same bin are similarly affected by the systematic process variation. Then for each bin, we train an NN-Watchdog. Besides, knowing that systematic variation impacts all devices within an IC similarly, the NN-Watchdog also provides a remedy to model this process variation.

4.3.2 Random Process Variation

To reduce the impact of random variation, we refer to the Central Limit Theorem, which establishes that averaging over a set with independent elements with zero-mean distribution will result in a normal distribution with a zero-mean, and a smaller variance equal to $\frac{1}{N}$ times the original variance [73]. This indicates that averaging, reduces the overall impact of process variation significantly.

To reduce the impact of random process variation in case of Trojan threat, using the

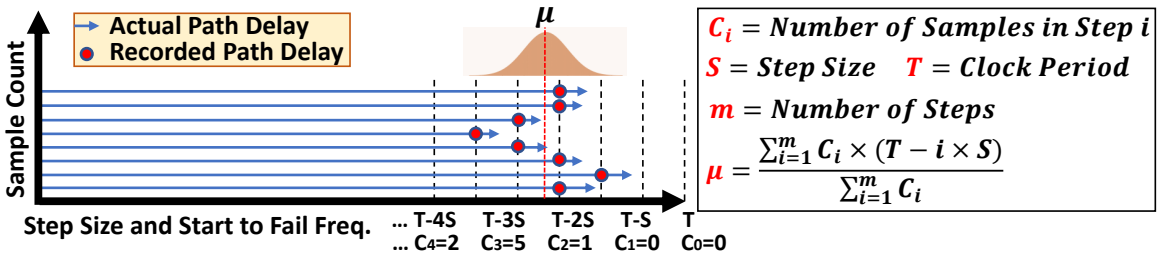


Figure 4.4: Computing the mean delay of a path using CFST delay measurements with step size S, clock period T, over m samples (dies).

formulation presented in figure 4.4, we collect the delay of each timing path (in our test set) from many ICs and compute their average delay to be used in our HW Trojan detection solution. When the timing-path delay is averaged across N different dies, the variance of the random variables representing the average delay is N times smaller than the variance of individual samples (${\sigma^2}_{AVG} = {\sigma^2}_{sample}/N$). Note that the mean value is computed from discrete delay samples obtained from CFST test, and the tester's size (S), as illustrated in figure 4.4, affects the value of the computed mean.

When the test targets recycled IC detection, our detection methodology relies on identifying the mean shift between the LAP and the MAP group, meaning we take the average of slacks of timing-paths in MAP group, and compare it with the average slack of timing-paths in LAP group. So in case of MAP group, the mean of added delay (AD) for timing paths in MAP can be denoted by random variable $\bar{X}$, where

$$\bar{X} = \frac{1}{n} \sum_{p_i \in MAP} X_i \tag{4.2}$$

With this information:

$$E(\bar{X}) = E(\frac{1}{n} \sum_{p_i \in MAP} X_i) = \frac{1}{n} E(\sum_{p_i \in MAP} X_i) = 0 \tag{4.3}$$

$$VAR(\bar{X}) = VAR(\frac{1}{n} \sum X_i) = \frac{1}{n^2} VAR(\sum X_i) = \frac{n\sigma^2}{n^2} = \frac{\sigma^2}{n} \tag{4.4}$$

The same analogy applies to the mean of AD computed for the LAP group. In another word, the mean shift (used for detection) is a 0-mean random variable with standard deviation $\sigma/\sqrt{n}$, where n represents the number of paths in MAP or LAP group. Therefore, by choosing a large number of paths (n) for each of LAP and MAP set (we collect thousands), the impact of process variation on mean-shift value becomes negligible, and process variation does not affect the final classification.

To emulate the systematic process variation (within the same process corner), we created 2 additional derivatives (slightly modified copy) for each of our skewed SPICE models. Each skewed SPICE model was altered to make the transistors in the first derivative 1% slower,

and in the second derivative 1% faster. To model Random process variation, each SPICE simulation is subjected to 100 Monte Carlo simulations (modeling CFST performed on 100 different dies in the same speed-bin), where the threshold voltage (V_{th}), Oxide thickness (T_{ox}) and channel Length (L) are varied (based on a normal distribution) to model the variation of path delays from chip to chip according to the expected variation in 32nm technology node.

4.4 Modeling Timing Impact of Voltage Noise

To improve the accuracy of our timing model (NGTM), we utilize a methodology [37] that models the voltage drop and voltage noise. The voltage modeling flow models the voltage drop and endpoint uncertainty (due to the voltage-induced clock jitter) using a differential voltage pair (different voltages for launch and capture path of a timing-path). The differential voltage pair is obtained based on a statistical analysis performed on the design-specific IR simulation results. By using this voltage modeling scheme, the voltage-induced clock jitter uncertainty becomes path specific. This removes the need for a large hard margin, resulting in the majority of timing-paths to benefit from the smaller and dynamically computed margins. We first introduce a new metric, coined as **Delay Equivalent Voltage (V_{DEV})** that could be used to express the effective voltage of a timing path. Then we illustrate how we can extract and use this metric to margin a design against IR drop and voltage noise, and illustrate how the computed IR drop and voltage noise generated from this flow track the physical and PDN changes.

4.4.1 Delay Equivalent Voltage

Consider the inverter-chain in figure 4.5. Each inverter, after physical placement, is connected to a different point of the on-chip PDN and experience a unique voltage signature. The Timing Window (TW) of a cell is defined as the time interval in which the cell propagates an arriving input signal to its output. *The supplied voltage to a cell can only affect*

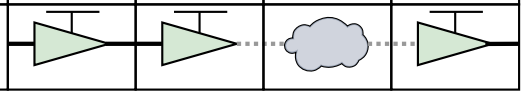
Model	Stage 1	Stage 2	• • •	Stage n
DEV delay model	(V_{DEV}, d_1)	(V_{DEV}, d_2)	• • •	(V_{DEV}, d_n)
Physical delay model	(V_1, D_1)	(V_2, D_2)	• • •	(V_n, D_n)
				

Figure 4.5: Inverter chain delay based on individual cell voltages when modeled by actual and V_{DEV} voltages.

the delay of a cell during its TW. Let's assume that during its TW the inverter at stage i experiences the average voltage V_i, and accumulated delay of the inverter and next stage wire when voltage V_i is supplied, is D_i. Hence, the total delay of the inverter chain is $\sum D_i$. The V_{DEV} is now defined as a single voltage that when applied to all inverters in the chain, the delay of the chain remains unchanged. In another word, the application of V_{DEV} changes the delay of inverter i from D_i to a new delay d_i, such that: $\sum_{i=1}^{N} D_i = \sum_{i=1}^{N} d_i$.

Using the alpha power model, the delay of a cell is defined as:

$$D_i \approx \frac{k_i V_i}{(V_i - V_{th(i)})^{\alpha}} \tag{4.5}$$

In this equation k_i is a technology dependent constant, V_i and $V_{th(i)}$ are respectively the voltage and threshold voltage of i^{th} cell, and α is the velocity saturation constant, where based on choice of technology node and process is bounded by $1 < \alpha < 2$ [74]. By differentiating this equation over voltage, the delay impact of small variation in the supplied voltage can be expressed as:

$$dD_i = -\frac{k_i - \frac{\alpha k_i V_i}{V_i - V_{th(i)}}}{(V_i - V_{th(i)})^{\alpha}} dV_i \tag{4.6}$$

Let's consider dD_i as the difference in delay of a cell when instead of V_i, the voltage V_{DEV} is applied. Additionally, let's define dV_i as:

$$dV_i = V_{DEV} - V_i \tag{4.7}$$

Based on its definition, when applying the voltage V_{DEV} to all cells in a logic path we expect the same delay (d_{path}) as of when each cell is annotated with its own unique voltage. More precisely, if the application of V_{DEV} to i^{th} cell in a logic path causes delay variation $dD(i)$, the overall path delay variation should be zero. Hence:

$$\Delta d_{path} = \sum_{i=1}^{N} dD_i = \sum_{i=1}^{N} -\frac{(k_i - \frac{\alpha k_i V_i}{V_i - V_{th(i)}}) \times (V_{DEV} - V_i)}{(V_i - V_{th(i)})^{\alpha}} = 0 \tag{4.8}$$

Let's define voltage headroom as $\Omega_i = \frac{V_i}{V_{th(i)}}$. After simplification and by using (4.5), we can rewrite the equation for V_{DEV} as:

$$V_{DEV} = \frac{\sum_{i=1}^{N} \frac{D_i[\Omega_i(1-\alpha)-1]}{\Omega_i - 1}}{\sum_{i=1}^{N} \frac{D_i[\Omega_i(1-\alpha)-1]}{V_i(\Omega_i - 1)}} \tag{4.9}$$

Based on this equation, the V_{DEV} of a timing path, by knowing the individual voltages of each cell could be calculated. In this equation, the $V_{th(i)}$ and V_i are known, and we only need the D_i, which is the delay of a cell when voltage V_i is applied. In order to compute the D_i quickly and effectively, we could use linear interpolation between delays specified in the library at different voltages. However, to improve the accuracy of the result, as illustrated in figure 4.6, we use non-linear interpolation between PVT corners defined using Composite Current Source (CCS) delay libraries [75]. In order to enable CCS non-linear interpolation, we need to have at least 3 CCS-enabled standard cell libraries at different voltages (for the same process and temperature). With this setup, we can generate timing sessions for each voltage within the range of voltages covered by CCS libraries. Using this setup, we perform timing analysis for multiple voltages in the desired range of IR drop and generate additional voltage-delay reference points. Now, let's consider that after IR analysis we obtain a cell voltage to be V_i. We first identify two closest timing sessions whose applied rail voltage

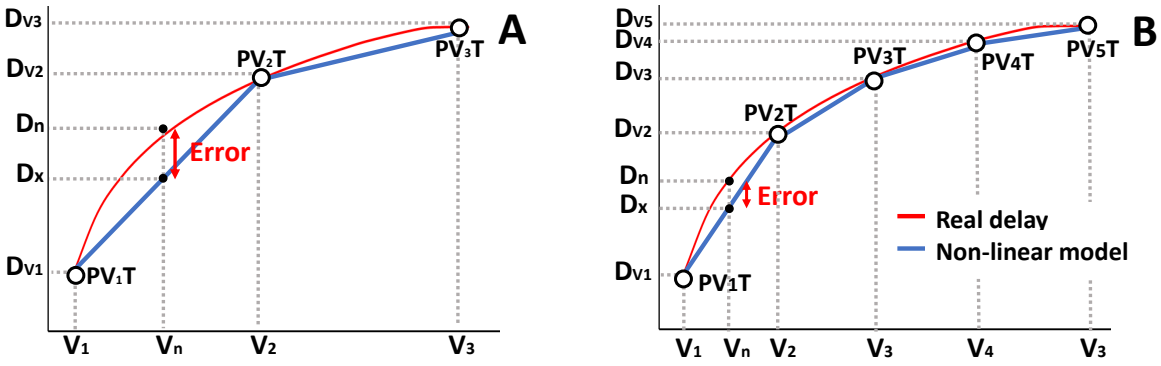


Figure 4.6: (left): Larger error for linear interpolation of a cell delay when using three timing session; (right): Generating two additional timing sessions using CCS non-linear interpolation followed by non-linear interpolation of the cell delay which resulting in a smaller interpolation error.

value encapsulates the V_i. Then we obtain the delay of the desired cell in each of the timing session. Let's denote the two timing sessions' voltages by V_{Slow} and V_{Fast} and the delay of the desired cell in each timing session by D_{Slow} and D_{Fast} respectively. The delay of the desired cell with voltage V_i could be obtained using the linear interpolation in equation 4.10. As illustrated in figure 4.6, by generating additional timing sessions (generated using CCS nonlinear interpolation), the error of the final linear interpolation for D_i is considerably reduced.

$$D_i = D_{Fast} + \frac{D_{Slow} - D_{Fast}}{V_{Slow} - V_{Fast}} \times (V_i - V_{Fast}) \tag{4.10}$$

4.4.2 Using V_{DEV} for STA annotation

In order to improve the accuracy of NGTM, we replace the IR drop and uncertainty hard margins, with a statistical representation of V_{DEV}, and bound the voltage of launch and capture sub-paths in each timing path separately. To do this, we rely on a recently supported feature of modern timing engines that support two different voltages for launch and capture paths when performing setup and hold timing checks.

The most contributing factor to clock Jitter is the dynamic change of voltage (voltage noise) from cycle to cycle. Hence, by providing a set of two different voltage for launch

and capture path, we can capture the worst case clock jitter effectively and remove the related endpoint register's uncertainty altogether. Let's consider the timing path in figure 4.7 during a setup check in two consecutive clock cycles; In the first clock cycle, the voltage of the cells in the common, and launch portion of clock path, leading to the launch register, will determine how fast the clock reaches the clock pin of launch register. In the second cycle, the voltage of the cells in the common and capture portion of the clock, determine how fast the clock signal reaches the capture register's clock pin. Considering that voltage changes from cycle to cycle, the arrival time of the clock to the launch and capture registers changes at each cycle. The worst-case arrival time of the clock, leading to the worst-case jitter is when the voltage in the first cycle is low (late lunch), and in the next cycle is high (early capture). Hence, if based on a dynamic IR drop analysis, we can provide the expected worst-case rail voltage values (the IR drop), and could statistically determine the worst-case change in rail voltage value from a cycle to the next, we could completely remove the uncertainty for the endpoint register, relying on the two (min and max) voltages to compute the worst-case jitter for each timing path. In this case, instead of using a fixed uncertainty value for the entire design, the amount of jitter is automatically computed for each timing path based on its launch and capture topology (the type and number of cells in each sub-path). Note that the original formulation for protecting the timing path against voltage noise jitter is overly pessimistic, as the degree of uncertainty was computed when worst

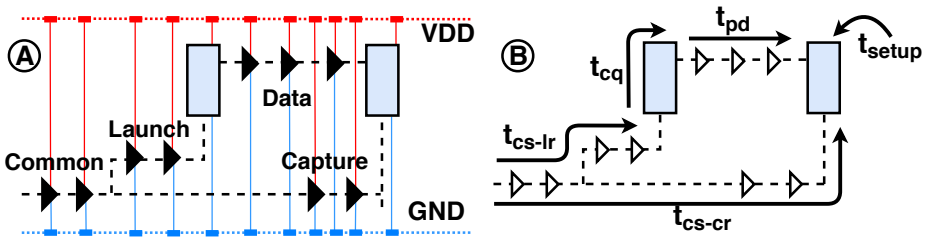


Figure 4.7: (left) The naming convention for different sections of timing path, (right): Delay components of a timing path

case voltage noise was applied to the worst case topology (longest clock path). However, in our proposed formulation, the impact of voltage noise on clock jitter is determined based on topology of each timing path, reducing the extent of pessimism by avoiding the double margining (worst case voltage noise + worst case path) against voltage noise.

In order to derive the proper voltages to perform the setup and hold check in a timing engine, we need to set up an IR simulation. For this, the IR drop is divided into that of the $package + die$, and that of the *board*. The IR drop in the *board* is of both resistive and inductive nature. However, the frequency of the RLC oscillation of the *board* is usually in the range of KHz to few MHz, which is much smaller than die frequency. Additionally, due to the large difference in their time constant (and frequency), the cycle to cycle variation on the die and RLC oscillation on the board could be considered as independent. Hence, using ANSYS Redhawk [76] we only simulated the $package + die$ and used the worst-case of IR drop in the board as a constant IR drop. The worst-case IR drop in the board, in typical industrial designs, based on the quality of board and DECAP engineering, is between 2% to 4% [77]. The package s-parameter model is then extracted and used in IR simulation. The starting voltage for IR analysis (at package balls) was then set to voltage regulator's voltage minus 4% drop in the board.

For IR simulation, due to time-consuming nature of IR analysis, we are constrained to perform the IR analysis for no more than a couple of hundreds of cycles. Hence, in order to capture the worst-case scenarios in our IR simulation (worst case IR drop and cycle to cycle voltage variation), we resort to the following methodology: For a given netlist, we find its max-power vector, and fast profile the power consumption across thousands of cycle using PrimeTime PTPX or Redhawk. From the obtained cycle-accurate power trace, we identify the following 5 scenarios: the part of the trace that contains (1) cycles resulting in the highest sustained power consumption (for at least 5 cycles), (2) cycles with largest increase in the power consumption from one cycle to the next, (3) cycles with largest drop in power consumption from one cycle to the next, (4) cycles with largest increase in the average power over two 10 cycles segments of the trace, and (5) cycle trace containing

the largest decrease in the average power over two 10 cycle segments of the power trace. These 5 scenarios are chosen to detect the worst case IR drop and worst case cycle to cycle voltage variation (possibly due to phase change in the input vector) of the design. Each of this scenarios is padded with 30 cycles of pre-simulation (10 of which is ignored when computing the V_{DEV} resulting in a total simulation of less than 200 cycles). By simulating a netlist for 200 cycles, we will obtain an effective voltage per cell per cycle. Using this simulation environment, the timing engine's voltages for launch and capture rail values could be computed as follows:

Using taxonomy in figure 4.7.(right) the $Common + Launch + Data$ portion was considered as Launch-Path (LP) and the $Common + Capture$ portion was considered as Capture-Path (CP). Using methodology described in section 4.4.1, we extracted the V_{DEV} for each of LP and CP of each timing path. Let $V_{DEV_L}(C_i, LP_j)$ be the V_{DEV} of the LP of the j^{th} timing path at cycle $i- > (i+1)$. The mean value of V_{DEV} across all cycles for all measured paths is obtained from:

$$\mu_{V_{DEV_L}} = \frac{1}{C \times P} \sum_{i=1}^{C} \sum_{j=1}^{P} V_{DEV_L}(C_i, LP_j) \tag{4.11}$$

In this equation, C is the number of simulated cycles; and P is the number of timing paths. The standard deviation of V_{DEV_L} is then used to capture the extent of launch voltage variation, which is obtained from:

$$\sigma_{V_{DEV_L}} = \sqrt{\frac{\sum_{i=1}^{C} \sum_{j=1}^{P} (V_{DEV_L}(C_i, LP_j) - \mu_{V_{DEV_L}})^2}{C \times P}} \tag{4.12}$$

To protect the circuit against aging effects such a Negative-Bias and Positive-Bias Temperature Instability (NBTI and PBTI) and Hot Carrier Injection (HCI), we could also include a measure $V_{Aging} = V_{NBTI} + V_{HCI} + V_{PBTI}$ to account for this phenomenon. At

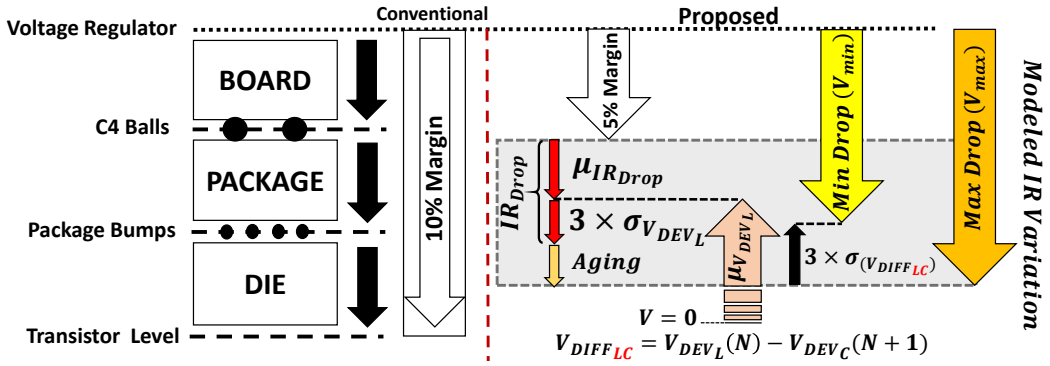


Figure 4.8: Modeling rail voltages, considering IR drop across board, package and die, for STA annotation.

this point, as illustrated in figure 4.8, we compute the voltage representing the largest drop on a sub-path (V_{max}) using:

$$V_{max} = \mu_{V_{DEV_L}} - K \times \sigma_{V_{DEV_L}} - V_{Aging} \tag{4.13}$$

Where K (e.g. K=3) is the guardband factor. We then, need to determine the extent of the voltage noise. The voltage noise of timing path P, from cycle i to cycle $i+1$ is obtained from:

$$V_{Diff}[C_i, P_j] = V_{DEV_L}[C_i, P_j] - V_{DEV_C}[C_{i+1}, P_j] \tag{4.14}$$

Considering P timing paths, and C cycles, there exist $P \times (C-1)$ data points for the voltage noise for all investigated timing paths. Using this data points the $\sigma_{V_{Diff}}$ and $\mu_{V_{Diff}}$, similar to equations (4.11) and (4.12) are extracted. Using these values, the voltage representing the maximum recovery from V_{max} within one cycle, denoted by V_{min} (minimum voltage drop) is obtained from:

$$V_{Min} = V_{Max} + 3\sigma_{V_{Diff}} + \mu_{V_{Diff}} \tag{4.15}$$

Note that V_{min} represents the maximum voltage departure of a timing path from V_{max}

that is achievable in a single clock cycle. *By using V_{min} and V_{max} as voltage values for LP and CP of a timing path, both IR drop and voltage noise are modeled. The voltage difference allows the timing engine to effectively compute the jitter per timing path and there is no longer a need for using an uncertainty margin for register endpoints for this purpose. Additionally, the jitter will be unique for each timing path depending on its topology, reducing the unnecessary jitter margin for the majority of timing paths.* Note that if we had input test vectors covering all worst scenarios, and we were able to simulate the IR drop for very large input vectors, we could have annotated each timing paths, based on its own distribution of V_{DEV}. However, in reality, for IR analysis and for practical purposes, we are limited to IR simulation for 100s of cycles. For this reason, we still need to statistically margin the voltage drop and voltage variation. However, in this case, the computed voltages (1) are computed based on realistic worst-case values observed in the design and not the rule of thump. (2) track the changes in PDN and physical design. Hence, not only the physical designer can safely reduce the margins to voltages suggested by proposed voltage modeling scheme, but could also observer the IR and timing impact of changes in PDN and physical design, and make more informed decisions. Algorithm 1 captures the process explained previously.

Algorithm 1 Computing V_{max} and V_{min} Rail Voltage Values

1: $\mu_{V_{DEV_L}} \leftarrow$ mean of $(V_{DEV_L}[Paths][Cycles])$; $K \leftarrow 3$;
2: $\sigma_{V_{DEV_L}} \leftarrow$ Standard deviation of $(V_{DEV_L}[Paths][Cycles])$;
3: $V_{max} \leftarrow \mu_{V_{DEV_L}} - (k \times \sigma_{V_{DEV_L}}) - V_{Aging}$;
4: **for all** C in $(Cycles - 1)$ **do**
5: **for all** P in $Paths$ **do**
6: $V_{diff}[P][C] \leftarrow V_{DEV_L}[P][C] - V_{DEV_C}[P][C+1]$;
7: $\mu_{V_{Diff}} \leftarrow$ mean of $(V_{diff}[Paths][Cycles])$;
8: $\sigma_{V_{Diff}} \leftarrow$ Standard deviation of $(V_{diff}[Paths][Cycles])$;
9: $V_{min} \leftarrow V_{max} + (K \times \sigma_{V_{Diff}}) + \mu_{V_{Diff}}$;

Usage of computed voltages is straightforward; State-of-the-art timing engines support dual rail voltages for LP and CP. For example, using Synopsys PrimeTime [78], the V_{max} and V_{min} could be applied using the following command:

$$PT > set\_rail\_voltage \quad -dynamic \quad -vmin < V_{min} > \tag{4.16}$$

$$PT > set\_rail\_voltage \quad -dynamic \quad -vmax < V_{max} > \tag{4.17}$$

Note that the proposed rail voltage modeling is far less pessimistic that annotating each cell with its worst case and best case voltage (as adopted by recent EDA tools) for the purpose of setup and hold timing check. This is because by using V_{DEV} we have accounted for the accumulated impact of delay variation due to individual cell voltage drops across all cells in a timing path. In addition, this voltage modeling technique accounts for the maximum voltage difference between the launch and capture portion of a timing path that **could be developed within one clock cycle**. Whereas if the cell voltage is annotated with their highest and lowest observed voltage (as adopted by several EDAs), such differential voltage is substantially exaggerated and is not based on physical reality, as such differential voltage could not be developed within one clock cycle.

4.5 Trojan Detection Flow

4.5.1 Detection Flow

Figure 4.9 shows the overall flow of the proposed Trojan detection flow. We augment the design stage with an additional step for statistical modeling of the voltage noise and IR drop using proposed voltage modeling. Accordingly, the STA reports the timing slack of each timing path based on its estimate of voltage drop and voltage noise (as opposed to a global pessimistic margin). This, as we will illustrate in the result section, will improve the correlation between timing slack predicted by timing engine, and the timing slack observed at test time using CFST. The final GDSII is then sent to the foundry for fabrication. The fabricated ICs may be tested in the untrusted foundry for functionality. The working ICs are then sent to a trusted facility for Trojan detection.

To detect a Trojan, we need to find the TT/TP induced slack change. As figure 3.1

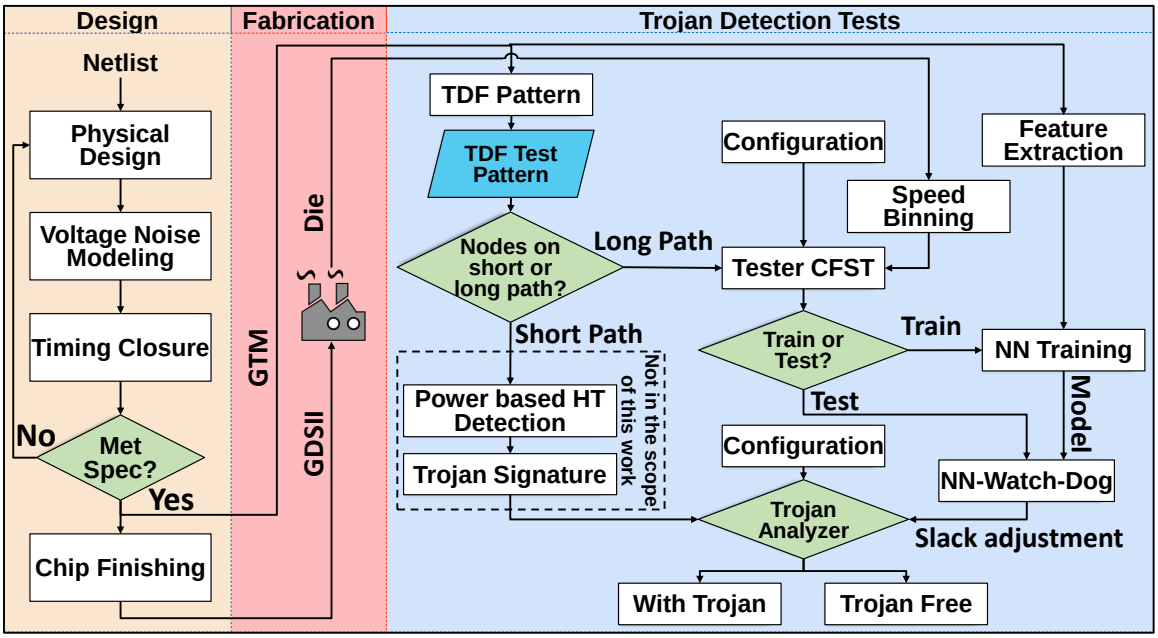


Figure 4.9: Trojan Detection Flow: The model includes changes in the design and test stages. The test stage divides the timing-paths into long and short paths. The short paths are subjected to power side-channel Trojan detection as described in [1] (not covered in this work), and the long paths are subjected to delay side-channel analysis using NGTM as reference timing model, adjusted by a NN that is trained as a process watchdog and by using CFST to find the start-to-fail frequencies for timing-paths under test.

shows, a TT adds capacitive load to driving cell of its *observed* net, and the TP appends an additional gate delay to every timing path that passes through its *victim* net. To detect a victimized or a monitored net (by a TP or TT), and for having no prior knowledge on which nets are affected, we need to include all nets in our delay analysis. We define a P2P-wire as a net that connects the output pin of a driver cell (or a primary input) to the input pin of one of its fanout cells (or a primary output). Hence a gate with a fanout of 4 has 4 P2P-wires. Each P2P-wire will be tested for rise and fall transitions. To increase the detection rate and to account for process variation, this process may be repeated for N different timing-paths passing through that net. The second criteria for selecting the timing-paths is the maximum frequency of the tester equipment; The delay of the selected paths should be larger than the limit imposed by the maximum reachable frequency of the tester equipment. If the P2P-wire in no timing-path is long enough for CFST, it is regarded as a candidate for Trojan detection via power-based detection schemes. Note that

timing-paths with a small number of gates (in their data sub-path) have high controllability, making them ideal for the power-based Trojan detection schemes (e.g. [12–14,45]) that rely on full or partial activation of such paths. For all other timing-path candidates, we generate the Path Delay Fault (PDF) test vectors using an Automatic Test Pattern Generation tool (ATPG). If ATPG cannot generate a test pattern for a path, the path selection changes. If ATPG cannot generate a test vector for any path through that P2P-wire, it is discarded.

Algorithm 2 Trojan Detection Flow

```
1:  N = # paths to be tested through each net in the design
2:  Nets ← all nets in the design.
3:  for all net in Nets do                          ▷ net selection of Path Delay Fault (PDF) test
4:      TimingPaths += select N timing-paths passing through net
5:  Perform speed binning on all dies and assign them to B bins.
6:  for all bin in B do                                                          ▷ NN training
7:      NN_bin ← Train a NN-Watchdog according to the algorithm 3
8:      σ_NN_bin ← the standard deviation of NN_bin
9:      for all die in bin do
10:         Slack = 0
11:         for all path in TimingPaths do
12:             CFST(bin,die,path) ← path slack measured by CFST die in the bin
13:             Slack(bin,path) += CFST(bin,die,path)
14:     for all path in TimingPaths do
15:         μ_S(bin,path) = Slack(bin,path)/sizeof(bin);
16:     T_Th= 4×σ_NN_bin                    ▷ Detection Threshold = 4σ to reduce false positive
17:     for all path in TimingPaths do
18:         GTM(path) ← query the slack of path from GTM
19:         NNSD(path) ← slack shift suggested by NN_bin(path)
20:         AS(path) = GTM(path) + NN_Watchdog(path)                          ▷ Adjusted Slack
21:         δ = μ_S(bin,path) - AS(path)                  ▷ Shifted delay after adjustment
22:         if (δ > T_Th) then                                                ▷ Trojan Classifier
23:             Likely Trojan Set ← path
```

Algorithm 2 describes our proposed Trojan detection flow. As described in this algorithm, after selecting the set of timing paths for PDF testing, we speed-bin the fabricated dies. In the next step, we collect the NN-Watchdog training data using the flow described in algorithm 3. Then, we train a process tracking NN-Watchdog for each bin and extract the standard deviation of each NN-Watchdog in predicting the shifted delays. For each bin, we perform CFST and measure the start to fail frequencies for the selected timing-paths. The slack difference (δ) between the mean of slacks reported by the CFST and the NN-Watchdog adjusted slack from GTM (in the same bin) represents the likelihood of a timing path being affected by a Trojan. To make a binary decision, we use a threshold to assess

the significance of δ and classify the timing paths into benign or malignant (Trojan) classes.

Algorithm 3 Generating a training set for the NN-Watchdog

```
1: NP ← mR^2                                   ▷ R is the registers count, and m is a large number (e.g. 10)
2: TimingPaths ← Select NP timing-paths (min of m path per endpoint)
3: for all path in TimingPaths do
4:     feature(path) ← Extract path features from GTM                          ▷ input feature
5:     GTM(path) ← Extract path slack from GTM
6:     Slack(path) = 0
7: for all die in Dies do
8:     for all path in TimingPaths do
9:         CFST(die,path) ← Slack of path in CFST test of die
10:        Slack(path) += CFST(die,path)
11: for all path in TimingPaths do
12:    Slack(path) = Slack(path)/NP;
13:    Δ_slack(path) = Slack(path) - GTM(path)                                 ▷ label
14:    data-points(path) ← (features(path),Δ_slack(path))
```

When choosing a value for Trojan-detection threshold, we face a trade-off between the false positive rate and the accuracy of Trojan detection. The false positive could be the result of 1) inaccuracy in the GTM, 2) inaccuracy of NN-watchdog, and 3) random process variation for sampling over a small number of ICs. To reduce the false positive rate, the threshold used for detection should be large enough, to account for these. Since we average the delay of each timing-path over many IC samples, the impact of random process variation in the average delay could be reduced to a desirable range. However, we still have to account for the inaccuracy of the NN and systematic variation. Hence, we define the detection threshold to be $T_{Th} = n \times max(\sigma_{NN}, \sigma_{processvariation})$, in which the $\sigma_{processvariation}$ is the expected variance of systematic process variation (excluding random) and σ_{NN} is the standard deviation of the NN. Since σ_{NN} is the aggregated impact of NN inaccuracy (for under-fitting or over-fitting of the trained model) and impact of systematic process variation, the variance of σ_{NN} tends to be larger than $\sigma_{processvariation}$, and we can simply use $T_{Th} = n \times \sigma_{NN}$ (n is selected as 4 in algorithm 2).

To verify the choice of threshold values T_{Th}, we utilized Youden[79] method to extract the threshold value from a Receiver Operating Characteristic (ROC) curve that we generate over our SPICE simulation data (details in chapter 5). *Note that at test time, we do not know which timing-paths are affected by HW Trojan. Hence, the optimal threshold of detection*

cannot be determined using the Youden method.

Change in the temperature affects the speed of transistors and alters the RC characteristics of the connecting wires. But, the temperature change is an extremely slow phenomenon. That's why one can design temperature sensors with sampling frequencies far lower than operational clock frequency [80,81]. At test time, a test vector is loaded into the scan chain using a slow clock, then the circuit operates at-speed for two cycles (launch and capture) using a fast clock. Finally, the scan is offloaded using a slow clock. The heat dissipation when using a slow clock is quite low, and the duration of at-speed test is only two cycles for each test pattern, limiting the extent of temperature changes to a fraction of a degree Celsius. Hence, at test time the die temperature can be tightly controlled to discount the delay impact of temperature variations.

To proceed with Trojan detection, as described in Algorithm 2, we first separate the fabricated dies according to their speed and performance into different speed-bins. Then, we train a process tracking NN-Watchdog for each bin and extract the standard deviation of each NN-Watchdog in predicting the shifted delays. In the next step, we perform CFST and measure the start-to-fail frequencies for different timing-paths. If the mean value of the slack difference between the slack reported by the CFST and the NN-watchdog adjusted slack from STA (in the same bin) is less than our detection threshold, the path classified as Trojan-Free.

A Trojan is detected if one of its TTs or TPs (Trojan signatures) is detected in the previous step. In the result section, in addition to reporting the rate of TT and TP detection, we also report the overall Trojan detection performance of our solution. In this case, a Trojan is detected if our proposed solution can detect at least one of its triggers or payload(s).

4.5.2 Diagnostic Analysis:

Our detection flow could classify a timing path as likely affected by Trojan for two reasons: 1) The timing path hosts the trigger or payload of HT, resulting in a true-positive, 2) the

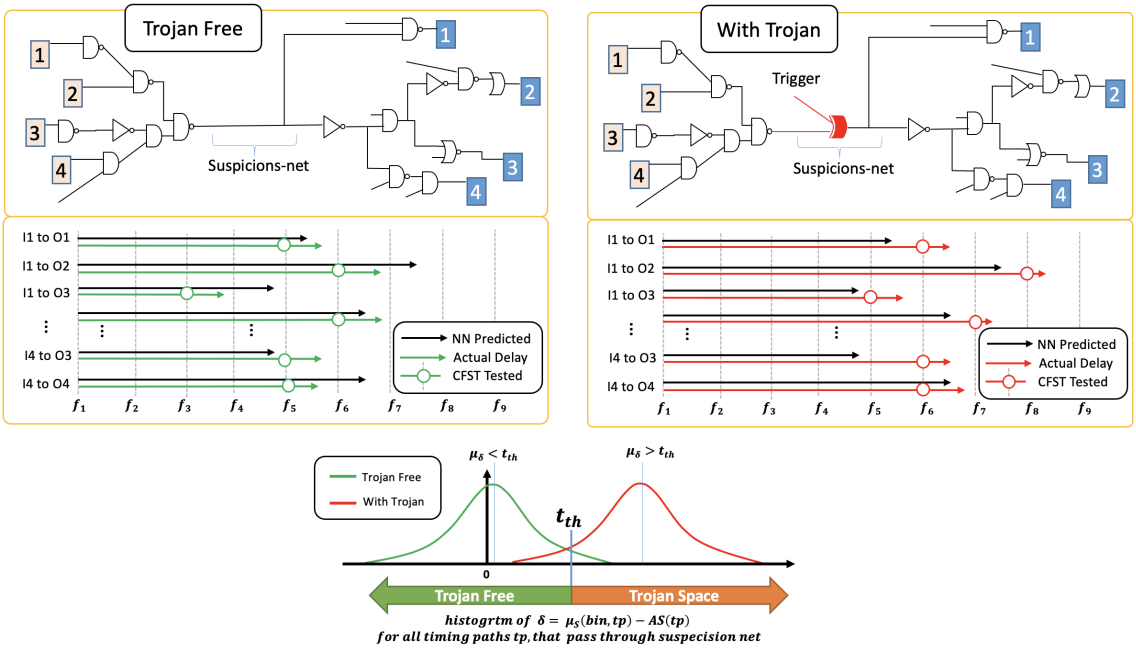


Figure 4.10: Diagnostic Test: (left): a Trojan free design where a suspicious net is tested for Trojan through many timing paths passing through it. The delta difference between predicted delay (NN adjusted STA delay) and CFST test delay (collected from multiple ICs) for each timing path is computed. In a Trojan-free design, the process variation results in a variation of the resulting delta value, but the mean of this delta difference is close to 0. (right): A design with HT on the suspicious net. The delta difference, in this case, is also a distribution. However, the existence of HT pushes the mean of this distribution away from 0. (bottom): The distribution of delta difference for the design with and without HT is shown. The Trojan is detected if the mean value of the delta difference distribution is greater than the detection threshold obtained from Alg. 2.

timing path is severely affected by process variation such that the resulting increase in the path delay is greater than the threshold, resulting in a false-positive. Our proposed diagnostic test opts to deeply investigate the detected HT and extract the reason behind the increase in each timing path's delay resulted in detecting such a Trojan. This diagnostic step's benefit is twofold: 1) reducing false positive, 2) pinpointing to the location of HT.

Algorithm 4 captures the flow of our proposed diagnostic solution. The overall strategy is quite simple and based on the following assumption: In a false positive case, the increase in the delay is due to process variation, and the total delay increase is distributed over different segments of the suspicious timing paths. Thus, by selecting N timing paths (that share least number of nets with the suspicious timing path whose net segments are being

Algorithm 4 Diagnostics

```
bin ← the speed bin that this IC is allocated to
NN_bin ← NN_bin trained for bin in Alg. 2
μ_S ← vector of mean values from Alg. 2
M_P ← Malicious Paths detected in Alg. 2
t_th ← Detection threshold from Alg. 2
n ← 50                                          ▷ number of new paths for diagnostics test
d_th ← threshold for diagnostics
for all path in MP do
    L_Nets ← list of nets in path
U = unique(L_Nets)                              ▷ remove repeated nets in the list
for all net in U do
    P_inc ← timing paths in M_P that include net net
    for all path in P_inc do
        L_exc ← all nets in timing path path except net
    L_test ← get n timing paths that include net and contains least number of nets in L_exclude
    for all tp in L_test do
        STA(tp) ← query the slack of tp from STA
        NN_watchdog(tp) ← slack shift from NN_bin(tp)
        AS(tp) = STA(tp) + NN_watchdog(tp)
        Δ += μ_S(bin,tp) - AS(tp)
    μ_δ = Δ / size(L_test)                      ▷ average shift over all paths
    if (μ_δ > t_th) then
        L_diagnostic append(net)
```

tested), we expect a small increase in the delay of N chosen paths compared to the NN-adjusted timing model prediction.

To run our diagnostic test, we select N timing paths that pass through the selected net for each net in the suspect timing path such that the selected timing path and the suspicious path do not include any common path segments other than the selected net. Then the delay of each of these timing paths is assessed against our NN-adjusted timing model. The delta δ between the average delay of that timing path reported by the CFST test and the delay of that timing path expected from our model is computed. We then compute the mean of δ (μ_δ) across all selected timing paths passing through that net. If the μ_δ is smaller than a threshold value (close to 0), then the CFST slack and adjusted slack for all timing paths through that net match (no mean-shift), indicating no Trojan and the net is removed from the suspicious list. However, suppose μ_δ is a positive number larger than the threshold. In that case, that indicates a mean-shift in the difference between predicted delay (NN adjusted delay) and recorded CFST delay across all timing paths passing through that net. This is an indication of a Trojan. In this case, the algorithm also pinpoints the location of the Trojan (the net where the means shift in the distribution of delays occurs). If it is not

possible to select N exclusive timing paths passing through a single net, a set of nets are selected to point to the HT location. Using this scheme, we reduce the false-positive rate, thus increasing the detection precision.

4.6 Recycled IC Detection Flow

Our approach consists of six main steps, described next:

- **ADP set identification:** Selecting a viable set of Age Distinguishing Paths (ADP), and dividing it into MAP and LAP subsets.
- **Building the Neural Assisted Timing Model:** Generating the NGTM model that accounts for process drift, timing prediction of which matches that of CFST test on MAP subset of timing paths. This step intends to model the impact of process drift and systematic process variation.
- **Computing Added Delays:** Inferring the slack of timing paths in the ADP set for both LAP and MAP subsets from the NGTM (created in the previous step) as expected value, and from CFST as actual value, and computing the $AD(p) = CFST(p) - NGTM(p)$ for each path in each subset.
- **Inferring MAP-LAP mean shift:** Computing the mean-shift of AD for MAP and LAP subsets; This step intends to reduce the impact of process variation and tester discrete step size on our detection threshold.
- **Classification:** Using a binary classifier to mark the IC as aged or new.

4.6.1 ADP set identification

Age distinguishing paths consist of two subsets of MAP and LAP timing paths. In order to collect the ADP set and assign timing paths to each of MAP and LAP, we propose the following set of modeling steps, each of which is described next:

- Train a regression model for gate-specific age perdition

- Build an Aging-Induced Path-Delay Prediction Model
- Build an ADP set Classifier

Regression model for gate-specific age prediction: To build a model that predicts aging-induced delay increase for each gate, we first create a database that includes each gate type and its corresponding aging-induced delay increase after i months when the gate is fed with different workloads. To emulate different workloads, the gate is simulated under different conditions where in each condition of $COND_{i,j}$, its output signal Duty Cycle and Toggle Count are DC_i and TC_j, respectively. Here, Duty Cycle (DC) denotes the percentage of the time that the signal is '0'.

In practice, we consider I different DC and J different TC values for the gate output, and for each condition (among all $I \times J$ conditions) we generate a table of input patterns (among many possibilities) that satisfies the considered toggle count and duty cycle on the output.

To tailor a more precise timing model for each circuit, we determine an approximate range of TCs that the circuit's gates outputs experience during run time by simulating the main circuit with a set of random inputs. We use the SAIF file which is generated via simulation to extract the maximum and minimum TCs of all signals in the circuit, refer to as TC_{min} and TC_{max} hereafter. Then, we sweep DC in range of $[0, 1]$ with the steps of st, and TC in range of $[TC_{min}, TC_{max}]$ with the steps of tc. In this study, st and tc are considered as 0.05 and $\frac{TC_{max}-TC_{min}}{50}$, respectively.

As the next step, the aging-induced delay change for each gate type in the considered TC and DC combinations are extracted via HSPICE aging simulations for i months using the FO4 model for each gate (as shown in figure 4.11, and are included in the database which is further deployed for training a *non-linear regression model* to predict the delay of each gate type experiencing unseen TC and DC combinations in its output.

Aging-Induced Path-Delay Prediction Model: Having the SAIF file used to specify the TC_{min} and TC_{max} and the regression model from the previous step, enables us to infer the

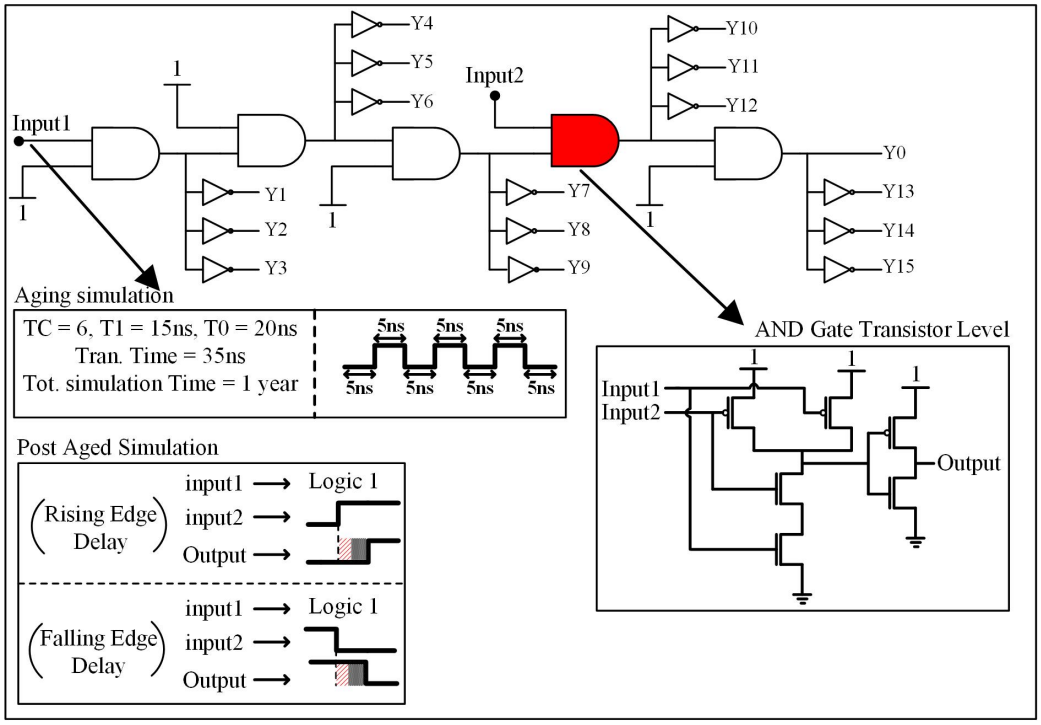


Figure 4.11: SPICE netlist for aging each gate type.

aging-induced delay change of each gate in the design after i months of aging, when the device experiences the switching activity and duration cycle in that SAIF file. Instead of building a new timing model, we use a trick to use existing STA engines for aging-induced path-delay prediction. For this purpose, we replace the delay of each cell, with the delay increment suggested by the regression model (developed in the previous step) and re-time the design. The timing report, in the result of query to the STA tool, provides us with the net delay increase of each timing path, accounting for possible skew in the launch and capture portion of each timing path (if they age differently or have different topology). At this point, we have all information needed to extract the ADP set, described next.

ADP set Classifier: The ADP set, is composed of MAP and LAP subset. Timing paths assigned to each subset have to satisfy two requirements. The first requirement that is common for both MAP and LAP groups is that the selected timing paths should have available slack s in the original (design time) timing model. Where s satisfies the inequality

$f_{max} > 1/(T - s)$, in which T is clock period, and f_{max} is the maximum clock frequency of the tester. The reason for this path selection is to be able to use CFST to measure the delay of the timing paths, which is needed as a part of our model building. The second requirement is to have a high value of aging-induced path delay prediction for paths in the MAP subset, and low values for timing paths in the LAP subset.

Identifying MAP and LAP subset could be easily achieved by plotting the histogram of aging-induced path delay predictions for all timing paths that meet the first condition ($f_{max} > 1/(T - s)$) and identifying set of timing paths that are (predicted to be) least and most affected by aging. This process could be automated by fitting a bimodal function on the resulting plot.

$$f = Gauss(\mu_{LAP}, \sigma_{LAP}) + Gauss(\mu_{MAP}, \sigma_{MAP}) \tag{4.18}$$

To differentiate MAP and LAPs from each other, we argue that a timing-path belongs to the MAP group if its aging-induced path delay prediction is greater than $\mu_{MAP} - 2 \times \sigma_{MAP}$; while it is included in the LAP group if it is less than $\mu_{LAP} + 2 \times \sigma_{LAP}$. Note that the timing paths with mid-range value for aging-induced path delay prediction (that extend the tail of MAP and LAP towards one another) are removed, and are mot included in ADP.

4.6.2 Building The Neural Assisted Golden Timing Model

The mean of aging-induced delay in LAP and MAP paths deviates from each other as the device ages. We utilize this observation to identify recycled chips. More precisely, the expected delay of timing path, as reported by STA, could be compared with the delay obtained from CFST: $AD = CFST(p) - NGTM(p)$. This AD could be due to the process variation, process drift (mentioned in section 3.2), and the impact of aging. To remove the impact of variation from this AD, we employ the NGTM by training the NN-Watchdog over the MAP subset of timing-paths. Having this setting, the new AD only represents the impact of aging.

4.6.3 Computing Added Delays

Now that we have the NGTM for the ADP set, we can compute the Added Delay (AD) of each path from:

$$AD(p) = CFST(p) - NGTM(p) \tag{4.19}$$

Note that this model is tuned to predict the delay of the MAP subset of ADP for the current chip, that could be aged or new. If the IC is new, the AD value for MAP and LAP subset should fall in the same distribution (when collected across many timing paths). However, if the IC is aged, the $Model(p)$ prediction (which is tuned for MAP subset) will be incorrect, resulting in large inconsistency between $CFSF(p)$ and $NGTM(p)$.

4.6.4 Inferring MAP-LAP mean shift

Assuming there are n paths in MAP subset, and m paths in LAP subset, we define the mean-shift MS as the

$$MS = \frac{1}{n} \sum_{p \in MAP} AD(p) - \frac{1}{m} \sum_{p \in LAP} AD(P) \tag{4.20}$$

4.6.5 Classification

The last step for detecting aged ICs is a classification based on a simple thresholding mechanism, Using threshold value Th, the IC is identified as aged when $Th \leq MS$. Choosing a value for Th introduces a trade-off between false positive and sensitivity of the test. The smaller Th the value, the more sensitive the test, and could even identify slightly aged devices at the expense of possible higher false-positive rate. In this work, we set the threshold to the step size of the CFST tester (to reduce the false positive rate), which we assumed to be 10ps. However, note that there are other mechanisms that could be justified for setting the threshold such as 1) goal-driven threshold to identify devices aged more than m months, 2) error-driven thresholds (such as $m\sigma$ of the error of neural network), 3) simulation-driven

thresholds based on the average change of delay of affected timing paths after m months of aging, etc. each of which could be justified based on the ICs use case.

Chapter 5: Results and Discussion

In this chapter, we first look at the improvements obtained by employing proposed voltage modeling, then we look at the accuracy of the NN-Watchdog in tracking the process drift, and then we present the result of applying our proposed test flow, for both Trojan and Recycled IC detection.

5.1 Proposed Voltage Modeling Accuracy

In this section, the accuracy of our flow in modeling the voltage noise and improvement in the timing closure are quantified.

5.1.1 Verification of Delay Equivalent Voltage

For the verification purpose, we used Ethernet, S38417 and AES128 netlists from IWLS benchmark suit [82]. Using Synopsys Design and IC Compiler, we first hardened these IPs using 32nm cell libraries. Then we run a dynamic vectorless IR simulation using Ansys Redhawk and extracted cell voltages for 100 cycles of dynamic simulation, padded with 10 cycles of pre-simulation using a toggle rate of 10% and 100% for data and clock cells respectively. Subsequently, we collected 4K timing paths from the routed design and calculated the V_{DEV} for each timing path. To accurately model the behavior of a timing path for setup timing check, the voltages applied to the launch and capture path should come from two consecutive cycles. For this reason, for each cycle of simulation, we have computed the V_{DEV} for each of launch and capture segment of each timing path for every cycle. In the SPICE simulation of a timing path, when its LP is annotated with the V_{DEV} of cycle n, the CP is annotated with V_{DEV} of cycle $n+1$. For the base case, we also annotate the LP and CP of each timing path with voltage observed in two consecutive cycles from Redhawk

analysis. Hence, when IR simulation for C cycles is available, the setup check could be constructed $C-1$ times. The hold check, on the other hand, uses the voltage of the launch and capture in the same cycle.

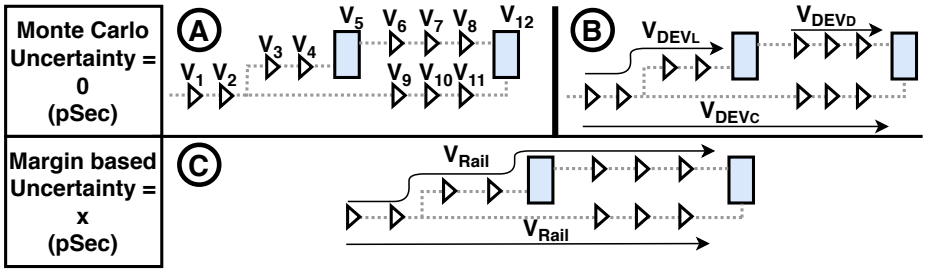


Figure 5.1: setup for a) the SPICE simulation when using actual voltages obtained from Redhawk; b) the SPICE simulation when using computed V_{DEV} voltages for LP and CP; c) the SPICE simulation when using hard margins (using 10% IR drop and 5% uncertainty)

Figure 5.1 (A and B) illustrate our setup for two sets of SPICE simulation, one when V_{DEV} is used and one when individual cell voltages are applied, and the resulting slack is compared. In order to further illustrate the accuracy of V_{DEV}, a third SPICE simulation is set up where the timing paths slacks are computed using the 10% IR drop rule for the cell voltages and 5% rule for the uncertainty. For verification, we computed the difference in the computed slacks when actual voltages are applied to that of when the modeled voltages (V_{DEV} or voltage obtained from 10% drop) is applied. In the SPICE simulation, the slack is computed using:

$$Slack = t_{cs-cr} + T_{clk} - t_{cs-lr} - t_{clk-q} - t_p - t_{setup} - U \tag{5.1}$$

where t_{cs-lr} and t_{cs-cr} are the delays of clock path from its source to the launch and capture register respectively, t_p is the longest propagation delay of data path, T_{clk} is the clock period, t_{clk-q} and t_{setup} are the inherent clock to Q, and setup delay for the launch and capture registers respectively, and U is the applied uncertainty. Using this equation, the slack differences are obtained from equations:

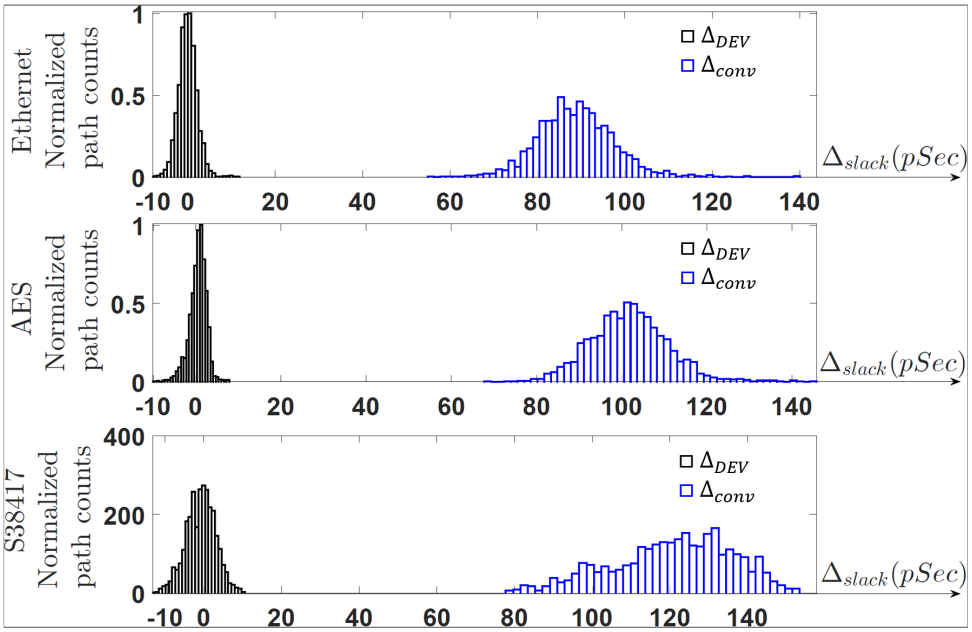


Figure 5.2: The timing slacks in three nearly timing closed design (Ethernet, AES128 and S38417) using conventional margin based and V_{DEV} flow for generation of NGTM

$$\Delta_{DEV} = |Slack_{Actual} - Slack_{V_{DEV}}| \tag{5.2}$$

$$\Delta_{Conv} = |Slack_{Actual} - Slack_{V_{Conv}}| \tag{5.3}$$

In this equations, $Slack_{Actual}$ is the slack obtained from the application of actual voltages in each cycle, the $Slack_{V_{DEV}}$ is the slack obtained from application of V_{DEV} values where uncertainty constraint is set to zero, and $Slack_{V_{Conv}}$ is the slack obtained by application of a rail voltage and voltage noise related uncertainty (similar to Synopsis PrimeTime). The histogram obtained from SPICE simulations of the timing paths for Δ_{slack} is illustrated in figure 5.2. As illustrated, for both benchmarks the Δ_{DEV} is a zero-mean distribution with much smaller standard deviation compared to the Δ_{Conv}. Smaller difference verifies the smaller error. Considering the Ethernet, S38417 and AES128 were designed for 1.4 GHz of frequency, the maximum path delay error for both benchmarks is reduced from $\sim$ 10% when using the conventional model, to around $\sim$ 1% when using V_{DEV}.

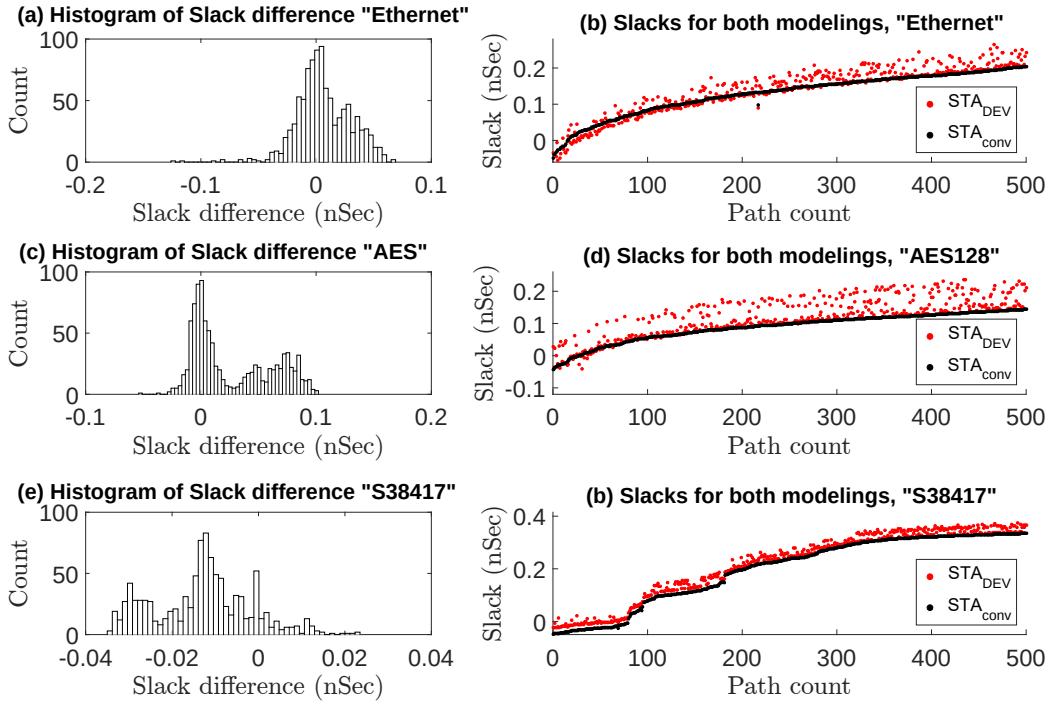


Figure 5.3: V_{DEV} based slacks v.s. margin-based slacks

5.1.2 Improvement in STA accuracy

To illustrate the timing impact of using rail voltages driven from V_{DEV} modeling flow, we performed a case study on Ethernet, S38417 and AES128 benchmarks. The STA was once obtained based on conventional flow (STA_{Conv}) and once using the proposed voltage modeling flow (STA_{DEV}) for IR drop and voltage noise.

Figure 5.3 illustrates the available slack for the critical timing path in STA_{Conv} and the recalculated slack based on STA_{DEV}. The slacks are sorted in ascending order as reported by STA_{Conv}. Hence, at each X location of this graph, the black dot represents the available timing slack based on the conventional hard-margin-based timing analysis flow, and the red dot represents the new timing slack obtained by using the proposed voltage modeling scheme. As illustrated, most timing paths see an additional timing slack, some as large as 100 pSec. From this graph, we can easily observe that the conventional flow, has penalized many timing paths with unnecessary margins. These margins, if available during physical

design flow, could be used for improving the Power, Performance and Area (PPA) of design by means of introducing additional V_T swapping or cell downsizing. In addition, in both benchmarks, there are several timing paths that are timing closed in STA_{Conv}, however, we see violation in STA_{DEV}, indicating that the original margins were not pessimistic enough. Hence, using STA_{DEV} could discover and fix this types of violations.

5.2 NN-Watchdog Accuracy

We evaluated the effectiveness of NN-watchdog on the three largest IWLS benchmarks [82] (Ethernet, S38417, and AES128). For training and test purposes, We collected a labeled dataset for each of these benchmarks using the methodology described in Algorithm 3 in section 4.5.1. We divided the dataset into a training and a test set with 80% and 20% of samples in each set accordingly. Besides, we evaluated the effectiveness of three regressors 1) Ridge, 2) MLP, and 3) Stacking-Regressor. Ridge is a linear regressor that has enhanced efficiency when it comes to parameter estimation problems. We use this model as a baseline. MLP, see Figure 4.2, is a non-linear regression. The stacking-Regressor (figure 4.3) is an ensemble of both linear and non-linear regressors, and represent our improved NN-watchdog model in this work.

Table 5.1 depicts the mean and standard deviation of the NN-Watchdog in predicting the shift in the delay of timing-paths when subjected to process drift. The process drift is represented using Fast, Typical, and Slow process corners which are simulated using skewed

Table 5.1: The Accuracy of Three NN-Watchdog regression model (Ridge Regression, MLP and Stacking-regressor) trained for different benchmarks on NGTM-10. The μ and σ are the Mean and Standard deviation of the regression error over the validation set. As discussed in Section 4.2, the Fast, Typical and Slow process are simulated using skewed Spice model with (X,Y) = (5,5), (0,0), (-5,-5), respectively. μ and σ are reported in pico seconds.

Benchmarks	**# Gates**	**Size Train/Test**	**Fast**						**Typical**						**Slow**					
			Ridge		**MLP**		**Stacked**		**Ridge**		**MLP**		**Stacked**		**Ridge**		**MLP**		**Stacked**	
			μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
AES128	114K	21K/4K	0.17	9.26	-0.14	7.45	0.09	5.16	-0.03	12.54	0.04	8.12	0.02	5.13	0.07	13.52	-0.02	7.15	0.01	5.14
Ethernet	40K	20K/4K	0.09	28.43	0.79	9.65	0.03	6.52	-0.12	28.82	0.28	9.13	0.01	6.51	0.08	29.43	-0.65	8.36	0.01	6.50
S38417	6K	4K/1K	0.07	12.41	0.12	6.87	0.09	5.03	0.17	13.93	0.08	7.07	0.09	5.02	-0.03	12.91	0.25	6.38	0.07	5.04

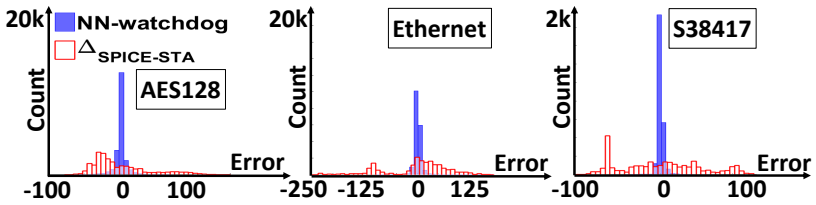


Figure 5.4: Histogram of NN-Watchdog Error trained for different benchmarks.

Spice model with (X,Y) = (5,5), (0,0), (-5,-5), respectively as discussed in Section 4.2. As shown, the standard deviation is reasonably small. To put this in perspective, we can compare the error distribution of NN-Watchdog with the error distribution obtained by finding the difference between delay of timing-paths reported by SPICE (d_{SPICE}) and that obtained from STA (d_{STA}). Figure 5.4 depicts the distribution of NN-Watchdog error and mean-shifted delay-difference model ($\Delta_{SPICE-STA} = d_{STA} - d_{SPICE}$) over a large selection of timing-paths. As illustrated, all models generate acceptable value for mean delay. The non-linear models (MLP and Stacked) result in a considerably smaller standard deviation across the board. As illustrated, the standard deviation in stacked regressor has also considerably improved compared to MLP, showcasing ensemble learning solutions' power in generating superior models. The lower standard deviation of the stacking-regressor model makes its prediction (of slack change) more accurate, resulting in a more precise detection solution, when integrated into our flow.

5.3 HW Trojan Detection Accuracy

5.3.1 Experimental Setup

We selected 720 timing-paths from non-critical to critical range, covering a range of 400 ps of slack from 3 largest IWLS benchmarks [82] (Ethernet, S38417 and AES128). Each benchmark is hardened and timing closed at 1.4 GHz in 32nm technology. For each benchmark, we divided the selected timing-paths into two groups (360 each) for inserting TTs and TPs. We further divided each subgroup into three smaller groups of 120 paths each to implement

small, medium, and large size Trojans. The TP size is controlled by the selection of logic gates with different inherent delays. The TT size is controlled by the distance it is placed from the triggering net. During NN-Watchdog training, we do not know if a timing-path selected for training contains a Trojan. Hence, we also evaluated the impact of including Trojans affected timing paths in the training; We trained 3 NN-Watchdogs with 0, 1, 5, 10 and 15 Trojan paths included in their training set. The rest of the Trojans are used for evaluating the proposed Trojan detection accuracy as a part of its test-set.

To model the voltage variation, we used Redhawk [76] and simulated 50 cycles of vectorless IR simulation when clock and data toggle rates are 100% and 10% respectively. In the SPICE simulation, each timing-path is assigned a random value from a normal distribution for the V_{th} of its transistors (to model the process variation), and each of its gates is annotated with the gate voltage reported by Redhawk in one simulation cycle. Note that each SPICE simulation presents a CFST test performed on a different die at a different time. Furthermore, the slack reported by the SPICE simulation for each timing-path was adjusted to the neighboring larger clock sweeping frequency step, modeling the CFST step size. The step size in the state-of-the-art tester equipment can be as small as 10-15ps. Hence, we selected the step size of the tester as 15ps.

In our simulations, we assessed the effectiveness of Trojan detection using two approaches. 1) Shifted STA (SSTA): when STA results are used as Golden Timing Model to detect HW Trojans. The process drift makes the direct usage of STA results quite ineffective. To account for process drift in SSTA, we have computed a static shift value, obtained from averaging the observed shift from many sampled timing-paths, and have shifted all reported slacks by STA using this value. For this approach, we have set the detection threshold to the fixed value of *45ps* which is the delay of a 2-input NAND gate in our standard cell library. 2) Neural shifted Golden Timing Model (NGTM) in which the voltage noise is modeled using V_{DEV} voltage modeling, while the process drift is modeled using NN-Watchdog. To show the stacked learning model's effectiveness, we have also evaluated the usage of both MLP and stacked-regression as NN-watchdog. When collecting a dataset

for training the NN-watchdog, there is no guarantee that the Trojan affected timing path(s) is not included in the training set. Therefore, we have investigated the accuracy of NGTM when the training set contains 0, 1, 5, 10, and 15 timing-paths affected by HW Trojans. In this approach, we have set our Trojan detection threshold to $4 \times \sigma$ of regressor standard deviation. The choice of $4 \times \sigma$ significantly reduces the number of false positives. Comparing the standard deviation of the stacked and MLP model, based on table 5.1, provides a valuable insight on why the NN-watchdog designed using the stacked-regression model is expected to be more sensitive/accurate compare to the MLP-regression model: It benefits from a lower detection threshold, while statistically benefit from a similar false-positive rate.

5.3.2 Trojan Detection Results

To evaluate the quality of selected threshold values for our proposed Trojan detection flow, we have extracted and reported the optimal threshold from the ROC-curve using Youden[79] method. Note the optimal threshold can not be extracted in real-life examples as it requires the ground-truth table (knowing exactly which timing path are and are not affected by Trojan), and could only be used for quality assessment purposes. The Youden method generates a different detection threshold for each of TT and TP based ROC curve.

Figure 5.5 captures the result of TP detection in Fast (X,Y)=(5,5) speed bin. The top row compares the accuracy of SSTA and NGTM in detecting TPs. The bottom row reports the false positive rate of detection for each model across different benchmarks. This figure evaluates Trojan detection's effectiveness when each of the Stacked-regression and MLP-regression models (for predicting the shift in slack) is used. The NGTM model is reported five times (NGTM-X), where each has been trained with X Trojans included in their training set, where $X \in [0, 1, 5, 10, 15]$. As reported, the inclusion of a small number of HW Trojan samples in our data set minimally impact the detection rate of Stacking model (using NGTM) on the test set, as the detection rate and false-positive rate of Stacking model for NGTM-0 is similar to the NGTM-X for $X \in [0, 1, 5, 10, 15]$. The similarity of detection rate and false-positive rate is simply because the number of Trojans is not statistically

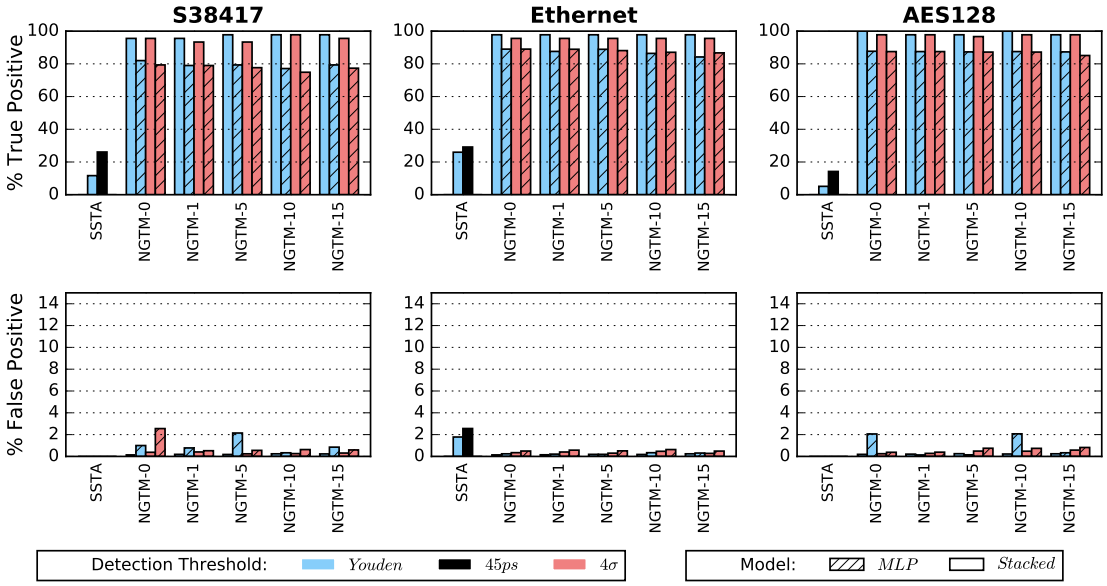


Figure 5.5: Trojan Payload detection results for 3 benchmarks. (top): Detection rate, (bottom): False positive rate. The SSTA bar represents the HW Trojan Payload detection using a (Mean shifted) STA. The NGTM bars represent the Trojan Payload detection when Neural-assisted timing model is deployed. Each bar shows the NN trained when X Trojans are included in the training set, with $X \in \{0, 1, 5, 10, and 15\}$.

significant to affect the training (e.g., 15 Trojan data versus 20K Trojan free data points) process.

The NGTM not only results in a significant increase in the TP detection rate (to over 95%) but also significantly depresses the false positive rate. This confirms NN-watchdog's ability in modeling the complicated, non-linear, path-specific shift of delays resulting from process drift. Finally, note that a small number of Trojans in the training set does not affect the accuracy of trained NN-watchdog as the impact of a few samples in a large training set is statistically insignificant.

Figure 5.6 depicts the result of our TT detection in the FAST speed bin with (X,X) = (5,5). Like the TP case, it compares the effectiveness of SSTA and multiple forms of NGTM (Trojan contaminated model) for detecting TTs. The figure shows how the improvement in the standard deviation of NN-watchdog significantly improves the detection rate for TTs (over 40% in some cases). As shown, NGTM has a lower rate for detecting TTs than TPs due to the smaller impact of TT on the delay of affected observed nets compared to TP (which

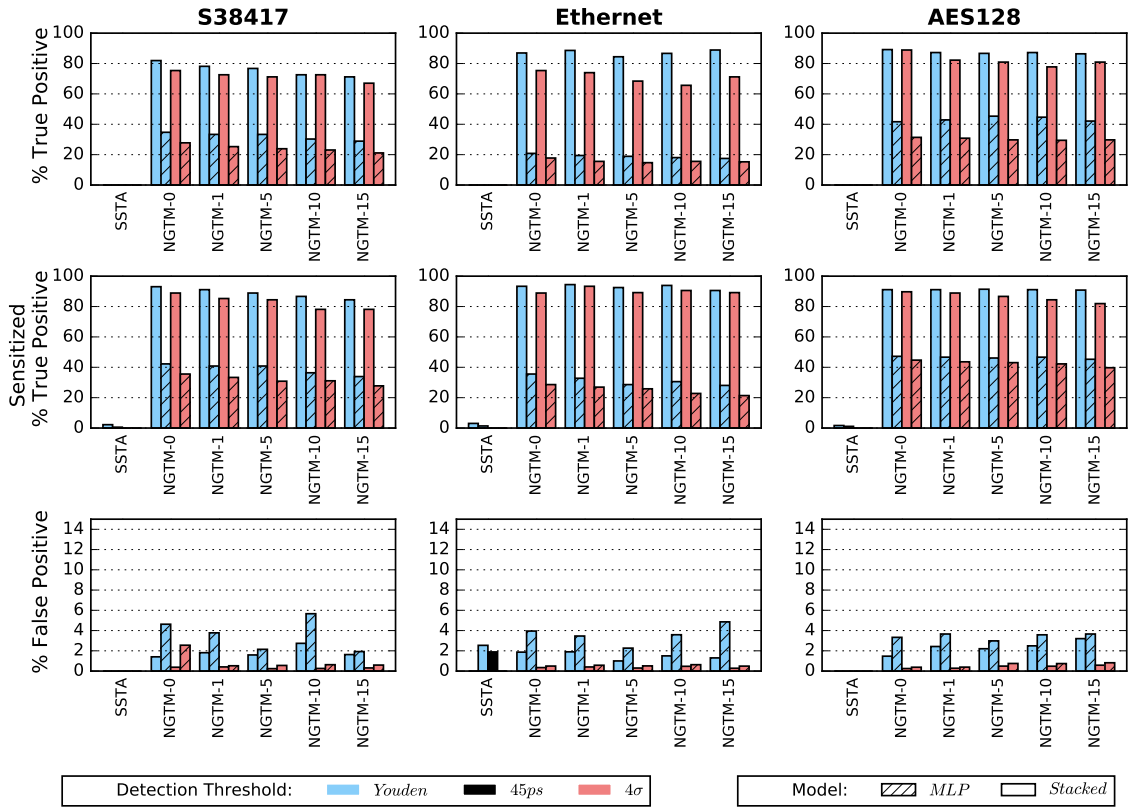


Figure 5.6: Trojan Trigger detection results for 3 benchmarks. (top): Detection rate, (middle): Detection rate for sensitized designs, and (bottom): False positive rate.

is at least equal to one gate delay). Like the TP case, we observe that contamination of the training set with few HW Trojan data points does not impact the accuracy of trained NN-Watchdog. As illustrated, the choice of the learning model (MLP vs. Stacked) for training the NN-watchdog has a significant impact on Trojan detection accuracy. As illustrated the stacking-regression, for having a smaller threshold (selected based on $4 \times \sigma$ of regression model error) could improve the detection rate by 10% to 15%, resulting in over 95% Trojan detection rate. This is when the false-positive rate of the overall solution, when constructed based on the stacking-regression model, is equal to or lower than its MLP-based counterpart. The selection of the Youden threshold for detection, although significantly improves the TT detection, results in higher false positive, and perhaps is not a preferred mechanism for setting the detection threshold.

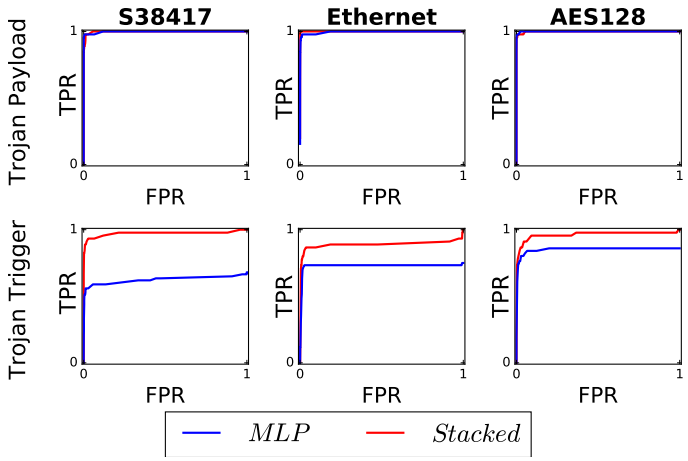


Figure 5.7: Associated ROC curve for (top): TP, and (bottom): TT, when NGTM-1 models are used. ROC curves capture the True Positive Rate versus False Positive Rate.

Figure 5.7 illustrates the ROC curve from which the Youden threshold (as described in Section 4.5.1) is extracted for NGTM-1. The threshold values used for detection using each of these methods is reported in table 5.2.

Table 5.2: Threshold values used for TT and TP Trojan detection in Fast-bin in Algorithm 2 when using NGTM-10 model

Benchmarks	**MLP**				**Stacking Model**			
	TP		**TT**		**TP**		**TT**	
	Youden	$4 \times \sigma_{NN}$	**Youden**	$4 \times \sigma_{NN}$	**Youden**	$4 \times \sigma_{NN}$	**Youden**	$4 \times \sigma_{NN}$
AES128	27.1	32.48	16.3	32.48	18.66	20.52	11.68	20.52
Ethernet	35.5	36.52	15.4	36.52	22.48	26.04	13.17	26.04
S38417	24.7	28.28	17.2	28.28	17.82	20.08	12.05	20.08

Table 5.3 captures the results of TP Trojan detection in all speed bins. As reported, the speed binning provides more accurate results for TP detection compared to the No-speed-binning case. This is due to the larger standard deviation of the NN-Watchdog when training over extracted delays from all dies without considering the impact of systematic process variation.

Table 5.3: Percentage of False Positives (FPos) and True Positives (TPos) when Stacking model , as described in Algorithm 2 is used to detect TP with different binning strategies (Slow, Typical, Fast, and no Binning). For this simulation, the NN is trained using a Trojan in dataset (NGTM-1 model).

Benchmarks	**Slow**		**Typical**		**Fast**		**No-Binning**	
	TPos	**FPos**	**TPos**	**FPos**	**TPos**	**FPos**	**TPos**	**FPos**
AES128	98.88	0.11	97.78	0.17	94.44	0.18	86.67	0.31
Ethernet	96.67	0.17	95.56	0.12	92.22	0.15	88.89	0.48
S38417	96.67	0.19	94.44	0.23	91.11	0.39	82.22	0.45

5.3.3 Results of Diagnostic Analysis

Our diagnostic test flow, as described in section 4.5.2, significantly reduces the false-positive rate of our Trojan detection solution. The result of running our diagnostic test is reported in Table 5.4. As shown in this table, the diagnostic test can significantly reduce the false positive rate, bringing it down to zero false positives for most NGTM, regardless of regressor choice (MLP vs. Stacking). Note that the choice of regression model determines our Trojan detection accuracy, and the diagnostic test flow does not help with increasing the detection accuracy. Hence, although our diagnostic test could impressively suppress the false-positive rate, the stacking learning model still has a clear advantage over the MLP model (or other liner models). In a few cases, the diagnostic test cannot reduce the false positive to zero. The reason for this observation is that the majority of timing paths selected for the diagnostic test are affected by process variation such that they see a slight increase in their delay, which is comparable to the Trojan Trigger impact. Besides, in some cases, the diagnostic test cannot generate enough timing paths that consist of a small portion of a malicious path, limiting the analysis on each segment (net) of a malicious timing path. To further reduce the false positive, one may a) increase the number of timing paths chosen for the diagnostic test, b) run the diagnostic test on a different die (if the Trojan is inserted in all dies), c) change the selection of timing paths used for diagnosis, or d) increase the detection-threshold for Trojan detection (trading off accuracy vs false-positive rate). We have purposely not repeated the experiment with additional timing paths to illustrate that the diagnostic test could still miss some of the false positives and highlight the need for

Table 5.4: In this table, the result of our diagnostic test for reducing the false-positive rate of our proposed model is reported. The diagnostic test is also able to pinpoint the location of nets hosting the Trojan Trigger or Payload. The expected number of suspect nets (by the model) after running the diagnostic test is indicated by E(n).

Benchmark	NN Model	# FPo Before Diagnostics	# FPo After Diagnostics	E(n)
AES128	NGTM-0	99	0	1.39
	NGTM-1	111	0	1.41
	NGTM-5	195	0	1.47
	NGTM-10	191	0	1.48
	NGTM-15	231	0	1.47
Ethernet	NGTM-0	137	0	1.61
	NGTM-1	161	0	1.62
	NGTM-5	119	0	1.65
	NGTM-10	187	3	1.65
	NGTM-15	111	2	1.68
S38417	NGTM-0	19	0	1.66
	NGTM-1	21	4	1.66
	NGTM-5	12	2	1.65
	NGTM-10	13	1	1.57
	NGTM-15	16	0	1.54

repeating the diagnostic test with one of the solutions proposed above.

Another advantage of our proposed diagnostic test, as previously suggested, is its ability to pinpoint the location (net) containing the Trojans' TT or TP. In some cases, however, we cannot produce enough test patterns for a single suspicious net, and we have to consider a set of nets for the diagnostic test. In such a case, the number of nets that should be diagnosed for Trojan is more than one. Table 5.4 summarizes the result of our diagnostic test. As indicated, the expected number of timing paths that may have a TT or TP Trojan, denoted by E(n), is between 1 and 2 timing paths. E(n) 's small value would significantly help with next-step verification for partially or fully-invasive Trojan detection (for verification) and save significant time scanning the IC for the Trojan.

5.4 Recycled IC Detection

We targeted 5 different IPs including s35932, s38417, s38584, b17, and AES128 from IWLS benchmark suite[82] and hardened them using a commercial 32nm technology via the Synopsys EDA toolset [83]. We used Synopsys HSPICE for the transistor-level simulations, and the HSPICE built-in MOSRA Level 3 model to assess the effect of NBTI and HCI aging [84]. The aging simulations were performed under temperature= 125°C and Vdd=0.85V for 12 months with 1- month steps.

For each benchmark, using the Synopsys PrimeTime tool, we extracted N=10 longest paths feeding each endpoint (flip-flop or primary output). To account for the tester frequency step size, in our experiments we only select a subset of these paths whose delay is at least 250ps, resulting in the selection of 3455, 2390, 2121, 2626, and 21460 timing paths for the s35932, s38417, s38584, b17, and AES128 benchmarks, respectively.

To take the impact of process variations into account, in our simulation-based setup, the random patterns we use to generate our NGTM is different from the set of patterns we use in aging simulations to extract the aging-induced path delays and creating the ADP classifier. Note that the ADP set classifier is unique for each GDSII netlist and is generated per design. Using the ADP set classifier, we fit a bimodal curve on ADP set's histogram for each design to identify MAP and LAP groups. The histograms and fitted bimodal curves for each target circuitry after one month of usage are shown in figure. 5.8. This figure clearly depicts the deviation of MAP and LAP paths from each other when the device is aged. This observation confirms the applicability of the proposed path classification scheme in detecting recycled chips.

As the next step after classifying the timing paths, for each benchmark, we extracted the features presented in table 4.1 from its GDSII file, and used them to generate 13 datasets per benchmark related to i months of aging where $0 \leq i \leq 12$. Each dataset includes the extracted features and the slacks collected from one of these 13 aging simulations. We used these datasets to generate a unique NGTM for each circuit-under-test.

We deployed the extracted NGTM for each benchmark circuits aged between zero and

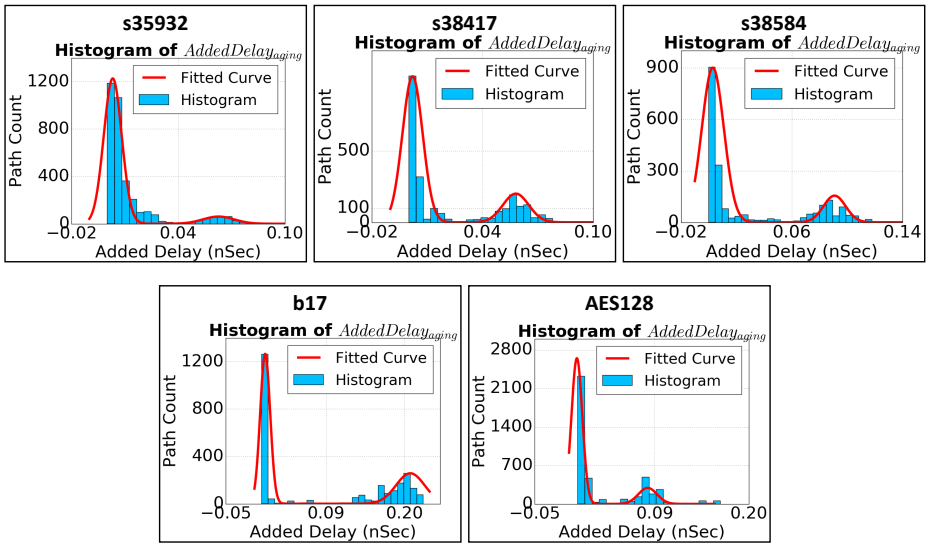


Figure 5.8: Histograms depicting delay-increase on timing-paths used for classification after one month of aging. For each benchmarks, there exists a bimodal distribution for the AD distinguishing the MAP and LAP paths from each other.

12 months, and calculated the MAP-LAP mean shift via equation 4.20. The results are shown in table 5.5. As depicted, the mean shift between LAP and MAP paths significantly increases for an aged device compared to its fresh (age=0) counterpart. The more the device is aged, the higher the value of the mean shift between its LAPs and MAPs. However, as expected the rate of mean-shift increase is higher initially. This is because the aging effect is high in the first couple of months but the aging-induced threshold voltage tends to saturate for long stress times (refer to figure 2.1). In particular, as table 5.5 shows for the s35932 benchmark, the mean shift changes from -0.63ps to 12.39ps (20x increase) after 1 month, while it increases 50% in the following month compared to its value in month 1. The same trend can be observed in other benchmarks. On average, over all benchmarks, the mean shift increases 29.15, 36.02, 44.19, 53.71, and 72.23 ps after 1, 3, 6, 9, and 12 months of aging. Accordingly, the proposed method can accurately differentiate the new and recycled chips from each other.

Table 5.5: The mean error of each ADP set group for all benchmarks.

Benchmark	Aging (Months)	0	1	2	3	4	5	6	7	8	9	10	11	12
s35932	**Train & Test (MAP)**	0.00	0.12	0.28	0.03	0.05	0.07	0.09	0.20	0.39	0.51	0.10	0.37	0.43
	Evaluate (LAP)	0.63	−12.27	−19.39	−22.32	−23.88	−23.87	−27.47	−28.54	−30.72	−30.91	−32.15	−35.71	−39.04
	Mean Shift	−0.63	12.39	19.67	22.35	23.92	23.94	27.56	28.74	31.11	31.42	32.25	36.08	39.47
	Correctly Identified	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
s38417	**Train & Test (MAP)**	−0.05	−0.89	−0.54	−0.76	2.81	−0.49	0.27	−0.20	−1.81	−0.96	−1.85	0.89	1.88
	Evaluate (LAP)	0.57	−28.93	−31.07	−32.76	−33.05	−33.55	−36.54	−40.91	−41.01	−43.60	−44.52	−49.99	−62.87
	Mean Shift	−0.62	28.04	30.53	32.00	35.87	33.06	36.82	40.71	39.20	42.64	42.67	50.89	64.75
	Correctly Identified	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
s38584	**Train & Test (MAP)**	0.00	−0.20	−2.54	1.40	−1.43	1.79	1.22	−1.05	−2.99	−0.30	−2.92	−2.86	−2.95
	Evaluate (LAP)	0.46	−26.46	−31.50	−34.47	−39.01	−41.02	−48.17	−46.42	−49.28	−49.53	−53.01	−55.15	−57.46
	Mean Shift	−0.46	26.26	28.96	35.87	37.58	42.81	49.39	45.37	46.29	49.23	50.08	52.30	54.52
	Correctly Identified	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
b17	**Train & Test (MAP)**	0.02	−2.75	−2.10	−5.46	−1.12	0.64	0.24	−1.05	−0.99	0.66	−0.42	−4.58	−4.21
	Evaluate (LAP)	0.14	−38.22	−40.65	−44.99	−48.21	−53.94	−57.18	−68.94	−73.40	−91.61	−100.83	−104.66	−107.85
	Mean Shift	−0.12	35.47	38.55	39.53	47.09	54.58	57.42	67.89	72.41	92.27	100.42	100.08	103.64
	Correctly Identified	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
AES128	**Train & Test (MAP)**	0.02	1.06	0.87	0.65	1.94	0.35	0.31	2.09	1.28	1.84	1.36	1.49	0.63
	Evaluate (LAP)	0.53	−35.31	−39.77	−42.55	−47.95	−44.71	−46.68	−45.18	−52.32	−51.52	−55.25	−62.85	−84.86
	Mean Shift	−0.50	36.37	40.65	43.20	49.89	45.06	46.99	47.28	53.60	53.36	56.61	64.35	85.49
	Correctly Identified	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Chapter 6: Conclusion

In this work, we presented a novel variation modeling and timing signature, and a promising methodology for Trojan and Recycled IC detection based on side-channel delay analysis, that does not require the availability and usage of a Golden IC. The proposed scheme relies on 1) improving the timing model at design time to account for voltage noise, and 2) training a Neural Network at test time that is used as a process tracking watchdog to model the process drift (while accounting for process variation). Our proposed solution does not rely on the existence of a Golden IC. We also presented a methodology to distinguish between two sets of timing paths that age more/less overtime.

We have reported that our Trojan detection flow could achieve close to 90% Trojan detection, and have shown a 100% Recycled IC detection in the selected benchmarks.

Bibliography

[1] M. Lecomte et al., "An on-chip technique to detect hardware trojans and assist counterfeit identification," *IEEE Trans. on VLSI Systems*, vol. 25, no. 12, pp. 3317–3330, 2017.

[2] A. Yeh, "Trends in the global ic design service market," *DIGITIMES research*, 2012.

[3] K. Z. Azar, H. M. Kamali, H. Homayoun, and A. Sasan, "Threats on logic locking: A decade later," in *Great Lakes Symposium on VLSI (GLSVLSI)*, 2019, pp. 471–476.

[4] H. M. Kamali, K. Z. Azar, K. Gaj, H. Homayoun, and A. Sasan, "LUT-lock: A Novel LUT-based Logic Obfuscation for FPGA-bitstream and ASIC-hardware Protection," in *IEEE Computer Society Annual Sympo. on VLSI (ISVLSI)*, 2018, pp. 405–410.

[5] S. Roshanisefat, H. M. Kamali, K. Z. Azar, S. M. P. Dinakarrao, N. Karimi, H. Homayoun, and A. Sasan, "DFSSD: Deep Faults and Shallow State Duality, A Provably Strong Obfuscation Solution for Circuits with Restricted Access to Scan Chain," in *VLSI Test Symposium (VTS)*, 2020, pp. 1–6.

[6] H. M. Kamali, K. Z. Azar, H. Homayoun, and A. Sasan, "SCRAMBLE: The State, Connectivity and Routing Augmentation Model for Building Logic Encryption," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2020, pp. 153–159.

[7] S. Roshanisefat, H. M. Kamali, A. Sasan, "SRCLock: SAT-resistant Cyclic Logic Locking for Protecting the Hardware," in *Great Lakes Symposium on VLSI (GLSVLSI)*, 2018, pp. 153–158.

[8] H. M. Kamali, K. Z. Azar, H. Homayoun, A. Sasan, "Full-Lock: Hard Distributions of SAT Instances for Obfuscating Circuits using Fully Configurable Logic and Routing Blocks," in *Design Automation Conference (DAC)*, 2019, pp. 89–94.

[9] ——, "InterLock: An Intercorrelated Logic and Routing Locking," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2020, pp. 1–9.

[10] G. Kolhe, S. M. PD, S. Rafatirad, H. Mahmoodi, A. Sasan, and H. Homayoun, "On custom lut-based obfuscation," in *Proceedings of the Great Lakes Symposium on VLSI (GLSVLSI)*, 2019, pp. 477–482.

[11] N. Karimi, J.-L. Danger, and S. Guilley, "On the effect of aging in detecting hardware trojan horses with template analysis," in *International Symposium on On-Line Testing And Robust System Design (IOLTS)*. IEEE, 2018, pp. 281–286.

[12] D. Agrawal et al., "Trojan detection using IC fingerprinting," in *Security and Privacy, 2007. SP'07. IEEE Symp. on*. IEEE, 2007, pp. 296–310.

[13] R. Rad, et al., "A sensitivity analysis of power signal methods for detecting hardware trojans under real process and environmental conditions," *IEEE Trans. on VLSI Systems*, vol. 18, no. 12, pp. 1735–1744, 2010.

[14] H. Salmani et al., "New design strategy for improving hardware trojan detection and reducing trojan activation time," in *IEEE Int. Workshop on Hardware-Oriented Security and Trust*, 2009, pp. 66–73.

[15] Y. Liu et al., "Hardware trojan detection through golden chip-free statistical side-channel fingerprinting," in *Proceedings of the 51st Annual Design Automation Conference*. ACM, 2014, pp. 1–6.

[16] C. Lamech et al., "Rebel and tdc: Two embedded test structures for on-chip measurements of within-die path delay variations," in *Proceedings of the International Conference on Computer-Aided Design*. IEEE Press, 2011, pp. 170–177.

[17] R. Rad et al., "Sensitivity analysis to hardware trojans using power supply transient signals," in *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*. IEEE, 2008, pp. 3–7.

[18] R. M. Rad et al., "Power supply signal calibration techniques for improving detection resolution to hardware trojans," in *2008 IEEE/ACM International Conference on Computer-Aided Design*, 2008, pp. 632–639.

[19] D. Du et al., "Self-referencing: A scalable side-channel approach for hardware trojan detection," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2010, pp. 173–187.

[20] K. Hu et al., "High-sensitivity hardware trojan detection using multimodal characterization," in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2013, pp. 1271–1276.

[21] Y. Liu et al., "Hardware trojans in wireless cryptographic ics: silicon demonstration & detection method evaluation," in *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2013, pp. 399–404.

[22] K. Xiao et al., "A clock sweeping technique for detecting hw trojans impacting circuits delay," *IEEE Design Test*, vol. 30, pp. 26–34, 2013.

[23] Y. et al., "Hw trojan detection using path delay fingerprint," in *IEEE Int. Workshop on HW-Oriented Security & Trust*, 2008, pp. 51–57.

[24] J. Li et al., "At-speed delay characterization for ic authentication and trojan horse detection," in *Int. Workshop on Hardware-Oriented Security and Trust*, 2008, pp. 8–14.

[25] Y. Jin et al., "Hardware trojan detection using path delay fingerprint," in *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE Int. Workshop on*. IEEE, 2008, pp. 51–57.

[26] X. Cui et al., "Hardware trojan detection using the order of path delay," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 14, no. 3, p. 33, 2018.

[27] I. Exurville et al., "Resilient hardware trojans detection based on path delay measurements," in *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2015, pp. 151–156.

[28] D. Ismari et al., "On detecting delay anomalies introduced by hardware trojans," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2016, pp. 1–7.

[29] U. Guin, K. Huang, D. DiMase, J. M. Carulli, M. Tehranipoor, and Y. Makris, "Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1207–1228, 2014.

[30] U. Guin et al., "On selection of counterfeit ic detection methods," in *IEEE North Atlantic Test Workshop*, 2013.

[31] M. Alam, S. Chowdhury, M. M. Tehranipoor, and U. Guin, "Robust, low-cost, and accurate detection of recycled ics using digital signatures," in *HOST*, 2018, pp. 209–214.

[32] GIPC. (2016) Counterfeits and their impact on consumer safety. "https://www.theglobalipcenter.com/gipc-testimony-to-u-s-senate-committee-on-the-judiciary-on-counterfeits-and-their-impact-on-consumer-safety/".

[33] N. Karimi, J.-L. Danger, and S. Guilley, "On the effect of aging in detecting hardware trojan horses with template analysis," in *int'l Symp. on On-Line Testing And Robust System Design (IOLTS)*, 2018, pp. 281–286.

[34] ——, "Impact of aging on the reliability of delay PUFs," *Journal of Electronic Testing, Theory and Application (JETTA)*, vol. 34, no. 5, pp. 571–586, 2018.

[35] F. Oboril et al., "Extratime: Modeling and analysis of wearout due to transistor aging at microarchitecture-level," in *Int'l Conf. on Dependable Systems and Networks*, 2012, pp. 1–12.

[36] N. Karimi, A. K. Kanuparthi, X. Wang, O. Sinanoglu, and R. Karri, "Magic: Malicious aging in circuits/cores," *ACM Trans. on Architecture and Code Optimization (TACO)*, vol. 12, no. 1, pp. 1–25, 2015.

[37] A. Vakil et al., "IR-ATA: IR annotated timing analysis, a flow for closing the loop between PDN design, IR analysis & timing closure," in *Asia and South Pacific Design Automation Conf.*, 2019, pp. 152–159.

[38] H. Li, Q. Liu, and J. Zhang, "A survey of hardware trojan threat and defense," *Integration*, vol. 55, pp. 426–437, 2016.

[39] N. Jacob, D. Merli, J. Heyszl, and G. Sigl, "Hardware trojans: current challenges and approaches," *IET Computers & Digital Techniques*, vol. 8, no. 6, pp. 264–273, 2014.

[40] 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 (TODAES)*, vol. 22, no. 1, pp. 1–23, 2016.

[41] M. Tehranipoor et al., "A survey of hw trojan taxonomy and detection," *IEEE Design Test of Computers*, vol. 27, no. 1, pp. 10–25, Jan 2010.

[42] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware trojan attacks: threat analysis and countermeasures," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1229–1247, 2014.

[43] S. R. Hasan, C. A. Kamhoua, K. A. Kwiat, and L. Njilla, "Translating circuit behavior manifestations of hardware trojans using model checkers into run-time trojan detection monitors," in *2016 IEEE Asian Hardware-Oriented Security and Trust (AsianHOST)*. IEEE, 2016, pp. 1–6.

[44] F. Wolff et al., "Towards trojan-free trusted ics: Problem analysis and detection scheme," in *Design, Automation and Test in Europe*, 2008, pp. 1362–1365.

[45] S. Wei et al., "Scalable hardware Trojan diagnosis," *IEEE Trans. on VLSI Systems*, vol. 20, no. 6, pp. 1049–1057, 2012.

[46] O. Söll, T. Korak, M. Muehlberghuber, and M. Hutter, "Em-based detection of hardware trojans on fpgas," in *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*. IEEE, 2014, pp. 84–87.

[47] H. Salmani and M. Tehranipoor, "Layout-aware switching activity localization to enhance hardware trojan detection," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 76–87, 2011.

[48] R. Rad, J. Plusquellic, and M. Tehranipoor, "Sensitivity analysis to hardware trojans using power supply transient signals," in *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*. IEEE, 2008, pp. 3–7.

[49] M. Potkonjak, A. Nahapetian, M. Nelson, and T. Massey, "Hardware trojan horse detection using gate-level characterization," in *2009 46th ACM/IEEE Design Automation Conference*. IEEE, 2009, pp. 688–693.

[50] D. Du, S. Narasimhan, R. S. Chakraborty, and S. Bhunia, "Self-referencing: A scalable side-channel approach for hardware trojan detection," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2010, pp. 173–187.

[51] M. M. Tehranipoor, U. Guin, and D. Forte, "Counterfeit integrated circuits," in *Counterfeit Integrated Circuits*. Springer, 2015, pp. 15–36.

[52] B. Shakya, U. Guin, M. Tehranipoor, and D. Forte, "Performance optimization for on-chip sensors to detect recycled ics," in *int'l Conf. on Computer Design (ICCD)*, 2015, pp. 289–295.

[53] (2013) G-19A test laboratory standards development committee, Fraudulent Counterfeit Electronic Parts; Avoidance, Detection, Mitigation, and Disposition. "https://www.sae.org/standards/".

[54] K. Huang, Y. Liu, N. Korolija, J. M. Carulli, and Y. Makris, "Recycled ic detection based on statistical methods," *TCAD*, vol. 34, no. 6, pp. 947–960, 2015.

[55] Y. Zheng, A. Basak, and S. Bhunia, "Caci: Dynamic current analysis towards robust recycled chip identification," in *DAC*, 2014, pp. 1–6.

[56] U. Guin, D. Forte, and M. Tehranipoor, "Design of accurate low-cost on-chip structures for protecting integrated circuits against recycling," *Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 4, pp. 1233–1246, 2015.

[57] K. K. Kim, "On-chip delay degradation measurement for aging compensation," *Indian Journal of Science and Technology*, vol. 8, no. 8, p. 777, 2015.

[58] S. Khan, N. Z. Haron, S. Hamdioui, and F. Catthoor, "NBTI monitoring and design for reliability in nanoscale circuits," in *Int'l Symp. on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, 2011, pp. 68–76.

[59] P. Vuillod, L. Benini, A. Bogliolo, and G. De Micheli, "Clock skew optimization for peak current reduction," in *Proceedings of the 1996 Int. Symp. on Low power electronics and design*. IEEE Press, 1996, pp. 265–270.

[60] S. Pant and D. Blaauw, "Static timing analysis considering power supply variations," in *ICCAD-2005. IEEE/ACM International Conference on Computer-Aided Design, 2005.*, Nov 2005, pp. 365–371.

[61] R. Ahmadi and F. N. Najm, "Timing analysis in presence of power supply and ground voltage variations," in *ICCAD-2003. International Conference on Computer Aided Design (IEEE Cat. No.03CH37486)*, Nov 2003, pp. 176–183.

[62] K. Arabi et al., "Power supply noise in socs: Metrics, management, and measurement," *IEEE Design & Test of Comp.*, vol. 24, no. 3, 2007.

[63] V. Wang et al., "A design model for random process variability," in *Int. Symp. on Quality Electronic Design*, 2008, pp. 734–737.

[64] K. Xiao et al., "A clock sweeping technique for detecting hw trojans impacting circuits delay," *IEEE Design Test*, vol. 30, pp. 26–34, 2013.

[65] M. Gruber, *Improving Efficiency by Shrinkage: The James–Stein and Ridge Regression Estimators*. Routledge, 2017.

[66] L. Breiman, "Stacked regressions," *Machine learning*, vol. 24, no. 1, pp. 49–64, 1996.

[67] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[68] T. Chen et al., "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd int'l Conf. on knowledge discovery and data mining*, 2016, pp. 785–794.

[69] H. Zou et al., "Regularization and variable selection via the elastic net," *Journal of the royal statistical society: series B (statistical methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[70] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[71] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[72] T. K. Ho, "Random decision forests," in *int'l Conf. on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.

[73] M. Rosenblatt, "A central limit theorem and a strong mixing condition," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 42, no. 1, p. 43, 1956.

[74] T. Sakurai and A. R. Newton, "Alpha-power law mosfet model and its applications to cmos inverter delay and other formulas," *IEEE Journal of solid-state circ.*, vol. 25, pp. 584–594, 1990.

[75] Synopsys. Composite current source delay modeling. Accessed July 10, 2019. [Online]. Available: https://news.synopsys.com/index.php?s=20295&item=122723

[76] ANSYS-Apache. (2020) Redhawk. Accessed Jan 10, 2020. [Online]. Available: https://www.ansys.com/products/semiconductors/ansys-redhawk

[77] E. Bogatin, *Signal and Power Integrity - Simplified*, 3rd ed. Prentice Hall, 2018.

[78] Synopsys. (2019) Primetime. Accessed July 10, 2019. [Online]. Available: http://synopsys.com/implementation-and-signoff/signoff/primetime.html

[79] N. Perkins et al., "The inconsistency of optimal cutpoints obtained using two criteria based on the receiver operating characteristic curve," *American journal of epidemiology*, vol. 163, no. 7, pp. 670–675, 2006.

[80] S. Chen et al., "Fully on-chip temperature, process, and voltage sensors," in *Proceedings of 2010 IEEE Int. Symp. on Circuits and Systems*, May 2010, pp. 897–900.

[81] M. Sasaki et al., "A temperature sensor with an inaccuracy of $-1/+0.8$ ° c using 90-nm 1-v cmos for online thermal monitoring of vlsi circuits," *IEEE Trans. on Semiconductor Manufacturing*, vol. 21, no. 2, pp. 201–208, May 2008.

[82] IWLS-org. (2005) Iwls 2005 benchmarks. Accessed July 10, 2019. [Online]. Available: http://iwls.org/iwls2005/benchmarks.html

[83] Synopsys. (2020) Synopsys toolset. Accessed April 10, 2020. [Online]. Available: http://synopsys.com

[84] Synopsys, "HSPICE User Guide: Basic Simulation and Analysis," 2016.

Curriculum Vitae

Ashkan Vakil received his B.Sc. degree in Electrical Engineering from the University of Mazandaran, Mazandaran, Iran. He received his M.Sc. degree in Electrical Engineering from the University of Bridgeport, CT, USA in 2015. He has worked earlier on Nanoelectronic, Carbon Nanotube conductivity, and energy harvesting via renewable resources.