Hardware Trojan Insertion in Finalized Layouts: From Methodology to a Silicon Demonstration

Tiago D. Perez and Samuel Pagliarini, *Member, IEEE*

***Abstract*—Owning a high-end semiconductor foundry is a luxury very few companies can afford. Thus, fabless design companies outsource integrated circuit fabrication to third parties. Within foundries, rogue elements may gain access to the customer's layout and perform malicious acts, including the insertion of a hardware trojan (HT). Many works focus on the structure/effects of an HT, while very few have demonstrated the viability of their HTs in silicon. Even fewer disclose how HTs are inserted or the time required for this activity. Our work details, for the first time, how effortlessly an HT can be inserted into a finalized layout by presenting an insertion framework based on the engineering change order flow. For validation, we have built an ASIC prototype in 65-nm CMOS technology comprising of four trojaned cryptocores. A side-channel HT is inserted in each core with the intent of leaking the cryptokey over a power channel. Moreover, we have determined that the entire attack can be mounted in a little over one hour. We also show that the attack was successful for all tested samples. Finally, our measurements demonstrate the robustness of our side-channel trojan against skews in the manufacturing process.**

***Index Terms*—ASIC, hardware security, hardware trojan (HT) horse, manufacturing-time attack, side-channel trojan (SCT), VLSI.**

I. Introduction

THE EVER-INCREASING cost to build high-end semiconductor manufacturing facilities—building a 3-nm production line is estimated to cost $15–$20B [1]—has made most design companies migrate to a fabless business model. In practice, fabless design houses can design and market integrated circuit (IC) solutions, but the actual IC fabrication is outsourced to a third party. This practice can potentially affect the trustworthiness of an IC as a foundry (or a *rogue element* within the foundry) can manipulate the design for malicious purposes [2]. Many fabrication-time threats have been studied recently [3]. For combating these threats, numerous techniques have been proposed for increasing the trustworthiness of an IC. Examples of these techniques are Split Manufacturing [4], Logic Locking [5], [6], [7], and IC Camouflaging [8]. Unfortunately, the current state of these techniques makes them unsuitable for large-scale production of ICs, either because of practicality [4] and/or insufficient security guarantees [9]. Thus, the current practices of a globalized supply chain leave the fabrication of ICs vulnerable to attacks.

Tampering an otherwise trustworthy IC can be done by inserting malicious logic or modifying specific aspects of the manufacturing process [11], [12]. These kinds of modifications are often referred to as hardware trojans (HTs). HTs are designed to leak confidential information, to disrupt a system's specific functionality, or even to destroy the entire system (referred to as time bomb). Various types of HTs have been studied recently [13], [14], [15], [16], [17], [18], [19], [20], [21], demonstrating the potential threat of this type of attack.

An IC's operating physical characteristics, such as timing, power consumption, electromagnetic radiation, and even sound, can be used as a side channel to indirectly reveal information that should be internal to the IC. For this reason, side-channel attacks (SCAs) often target keys of embedded crypto cores [22]. However, to mount a successful SCA, acquiring a large amount of data is usually required, followed by correlation/statistical analysis. Moreover, a very specific type of HT has been proposed for assisting SCAs. Lin et al. [13] were the first to propose an HT architecture for assisting a power SCA, referred to as malicious off-chip leakage enabled by side-channels (MOLES). This specific type of trojan is the centerpiece of our work; in the remainder of this text it is referred to as a side-channel trojan (SCT). By using SCTs, the attack time can be drastically reduced as no further processing is required. The disadvantage of SCTs is their invasive nature. Inserting an SCT requires a modification of the circuit at fabrication time. While this might seem a difficult task at first sight, we later show how it can be executed by a capable attacker.

Despite encouraging results reported from the SCT studies, only Perez et al. [10] discussed how SCTs could be inserted *from the perspective of the attacker*. Even more concerning, this discussion is also absent from silicon-validated trojans [15], [16], [21]. In this work, we present an extension of the SCT design methodology described in [10]. We assume that a rogue element inside the foundry is the adversary and that he/she makes use of *readily available engineering change order* (ECO) capabilities of physical design tools. Consequently, the main contribution of this work, in addition to a framework for inserting SCTs, is an ASIC prototype for validating our methodology. A rich discussion regarding the effectiveness of the approach is also provided.

Manuscript received 17 June 2022; revised 19 August 2022 and 17 October 2022; accepted 16 November 2022. Date of publication 21 November 2022; date of current version 20 June 2023. This work was supported in part by the Project "ICT Programme" which was supported by the European Union through the European Social Fund and in part by the Estonian Research Council under Grant MOBERC35. This article was recommended by Associate Editor J. Rajendran. *(Corresponding author: Tiago Perez.)*

The authors are with the Department of Computer Systems, Tallinn University of Technology, 19086 Tallinn, Estonia (e-mail: tiago.perez@taltech.ee; samuel.pagliarini@taltech.ee).

Digital Object Identifier 10.1109/TCAD.2022.3223846

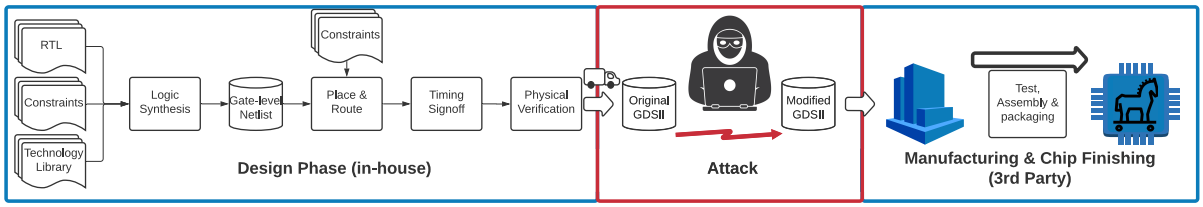

Fig. 1. Typical IC design flow. Highlighted in red is the stage where a rogue element may mount an attack (modified from [10]).

Furthermore, we have fabricated a test chip comprising four cryptocores in a 65-nm commercial technology, validating our technique in silicon. Our in-depth analysis of the side-channel reading versus the variation in the manufacturing process demonstrated that our SCT is robust against process variation. On top of that, we include an analysis of the attack time required for inserting our SCT in a finalized layout when also utilizing our ECO-based methodology. These characteristics sharply contrast our work with the prior art.

In Section II, we describe our considered threat model and the attacker capabilities. Our SCT architecture and our insertion methodology are detailed in Section III. In Section IV, we present our experimental results. In the first part of Section IV, we assume the role of the victim by implementing the target circuits. Later, we assume the role of the adversary by designing and inserting the SCT, utilizing the methodology described in Section III. In Section V, we present our prototype and describe the experiments executed during its validation. Our results discussion, along with a thorough discussion regarding HT detection, are presented in Section VI. Finally, we draw our conclusion in Section VII.

II. Threat Model and Attacker Capabilities

In this work, the principal attacker we are concerned with is a *rogue element* inside the foundry. His/her aim is to insert malicious logic into a finalized layout. We emphasize that the attack occurs before the fabrication, and a single rogue element inside the foundry is sufficient to perform the proposed attack. Thus, since the attacker is located inside the foundry, he/she enjoys access to all technology and cell libraries[1] utilized by the victim when creating the layout. We assume the attacker can identify the presence of a crypto core in a layout, which is a reasonable assumption, especially for well-known AES implementations that display regularity (due to the round-based key schedule structure). To be very clear, we do not assume that the adversary understands the entire victim's design (nor is there a need for such knowledge). Instead, we assume that the adversary can recognize the layout/structure of a single crypto core within a larger design, in line with the assumptions made in [14] and [16].

Furthermore, we also assume the adversary: 1) is versed in IC design; 2) enjoys access to modern EDA tools; and 3) has no means to make radical modifications to the circuit (e.g., adding new IOs or making changes in the clock domains). With the help of the inserted logic in the form of an SCT, the attacker will then attempt to leak confidential information via a power signature. Crypto cores are often the target in this type of attack [16], [17]—this is also the case in our work. As our attack deals with power signature reading, stopping some part of the clock delivery, or even entirely, would be highly beneficial for the attack. However, the attacker is assumed to have no knowledge about the clock domains or clock distribution in general. Synchronizing and controlling the HT's trigger to totally stop the clock delivery is not an option we have considered feasible, nor is the addition of an external trigger controlled by an IO.

A typical IC physical implementation flow is described in the left portion of Fig. 1. The attack takes place after the victim's layout in GDSII format is sent for fabrication (see a red portion of Fig. 1). Suppose the attacker had access to all of the victims' data required to generate the layout (i.e., RTL, netlists, constraints, etc.). In this case, he/she could replicate the physical implementation flow to achieve a layout similar to the one created by the victim, yet now containing his malicious logic. This effort is theoretically possible but largely unpractical. Our threat model, therefore, assumes that the attacker only has access to the finalized layout. Design companies have to hand over their finalized layout to the foundry for fabrication. Normally, the layouts require some preprocessing steps before the start of the fabrication, which are handled by a foundry employee. Thus, it is during this time period that the attack can be mounted.

Nevertheless, EDA tools already have the capability to deal with finalized designs. This functionality is a feature referred to as ECO. Thus, an attacker holding only the layout could use an ECO to modify or insert additional logic in a finalized layout. An ECO flow requires four inputs: 1) a technology library; 2) a cell library; 3) the gate-level netlist; and 4) a timing constraint. The adversary already possesses the first two, but must generate the third and estimate the fourth input. A gate-level netlist can be extracted from the victim's layout [23], while the timing constraint can be estimated to a certain degree. Our proposed trojan insertion framework is shown in Fig. 2, where these two steps are considered. The details of the framework are described in the next section.

III. Side-Channel Trojan Design and Insertion

A. Side-Channel Trojan Design

Our SCT is an additive HT with the intention of aiding a SCA. For more details regarding HTs taxonomy and concepts we direct the reader to [24]. Thus, it is designed for creating

[1]This is particularly true for advanced nodes where only a handful of cell libraries per node exist. Typically, the foundry or a company licensed by the foundry provides a standard cell library. In either case, we assume the attacker has no difficulty identifying individual gates and their functionality.

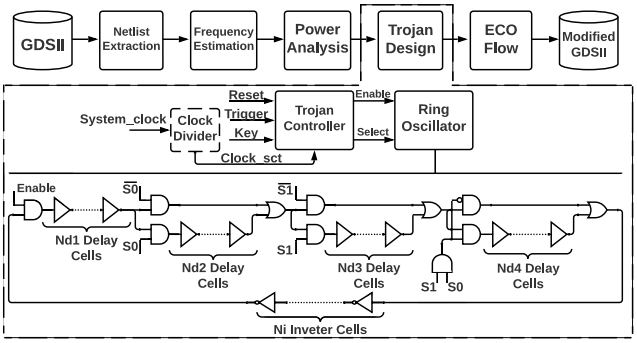

Fig. 2. Our trojan insertion methodology for an SCT capable of leaking 2 bits per power signature reading (modified from [10]).

TABLE I
RING OSCILLATOR ACTIVE PATH CONFIGURATION

S0	S1	Delay Cells	Inverter Cells	Freq.
0	0	N_{D1}	N_i	High
1	0	$N_{D1} + N_{D2}$	N_i	Mid-high
0	1	$N_{D1} + N_{D3}$	N_i	Mid-low
1	1	$N_{D1} + N_{D2} + N_{D3} + N_{D4}$	N_i	Low

an artificial power consumption through which information is leaked. This has to be performed in a controlled manner, naturally. Knowing that the majority of the power consumption in a circuit comes from the switching activity (dynamic power), a great candidate to be a controlled power sink is a structure with a controllable frequency of operation.

A ring-oscillator (RO) is an example of such power sink if the number of stages in the RO can be adjusted dynamically as shown in Fig. 2. Our architecture implements variable delay stages broken into branches that are controlled by N_{leak} leaking bits. Each branch of our RO has two active path options: 1) a direct connection to the next branch or 2) a series of delay cells. Thus, each set of the N_{leak} is associated with a distinct change in the power consumption amplitude. This artificial power consumption created by the RO is similar to a pulse-amplitude modulation technique, with an order equal to $2^{N_{\text{leak}}}$. An example of this RO architecture for $N_{\text{leak}} = 2$ is illustrated in Fig. 2. The active paths' configuration is described in Table I, where the leaking bits become branch selectors and are referred to as S0 and S1.

A dual-sided constraint guides the attacker's effort: he/she has to induce a discernible amount of dynamic power (i.e., to increase the effectiveness of the attack) while increasing leakage power as little as possible (i.e., to avoid detection). In this sense, not only the RO-based SCT structure has to be carefully planned, but a decision has to be made as to when exactly will the trojan be triggered. Our approach is to not allow the trojan to compete with the dynamic power consumption of the crypto core. Therefore, when the core is actively working, the trojan is silent and the RO is not switching. When the crypto core is idle, the trojan is triggered. For this reason, our proposed SCT trojan has a Trigger signal that is connected to the "done" signal coming from the crypto core, which marks the end of a cryptographic operation.

When triggered, the SCT connects a set of the leaking bits per clock cycle in the RO until all the N_{key} bits from the crypto key are leaked. Thus, our SCT requires a connection to the system clock and reset, a trigger signal, and the crypto key. Its architecture is illustrated in Fig. 2, consisting of three blocks: 1) clock divider (DV); 2) the trojan controller (TC); and 3) the RO. Notice that our SCT does not required any additional external connections (i.e., I/Os or pads). All of its signals are connected to existing wires of the target circuit. The invasive portion of our attack is the insertion of new cells and routing wires. Thus, the original logic of the target circuit remains unaltered.

The DV is responsible for dividing the frequency, as the name suggests. This feature is interesting for two reasons: there are scenarios when the attacker wants to slow down the speed at which bits are leaked, thus giving him/her control over the amount of time the attack will take. In other scenarios, the crypto core operates at a frequency that the TC cannot match, so to avoid timing violations in the HT itself, clock dividing proves useful. Thus, the clock_sct signal is either connected directly to the system_clock or to the DV. The TC is responsible for enabling the RO and for connecting the leaking bits to the RO. The RO starts running when the enable signal is asserted; its operating frequency is controlled by the select signals S0 and S1 (see Fig. 4).

To reduce the detection probability and increase the attack's feasibility, the SCT has to be customized for each targeted circuit. Therefore, the SCT is designed with size and power constraints. The size constraint is set as a percentage of the target circuit size. The power constraint is a percentage of the target circuit's idle power. As the size and power constraints are set by analyzing the circuit's physical characteristics, the attacker has to acquire such information from the layout. According to the flow detailed in Fig. 2, the layout is inspected as follows.

1) *Netlist Extraction:* Since the attacker only holds the layout, a gate-level netlist has to be extracted. Such effort is considered a trivial task for an expert IC designer, demonstrated in [23].
2) *Frequency Estimation:* The attacker has to estimate the operating frequency of the target circuit by performing static timing analysis on the extracted gate-level netlist. The attacker can observe the critical path(s) and then increase/decrease the frequency as needed to make the timing slack positive but near zero.[2]
3) *Power Analysis:* With the extracted gate-level netlist and the estimated frequency, the attacker can perform a typical power analysis. For relatively large circuits, static power can be estimated very precisely even without input vectors.[3]

Therefore, after the layout is inspected, the attacker has acquired the estimated frequency, estimated power consumption, and exact size of the circuit (number of gates). From

[2]Currently, our method does not take into account multicycle and false paths, which can reduce the accuracy of the frequency estimation but does not prevent the attack.

[3]For crypto cores in particular, it is a fair assumption to consider the plaintext to be randomly assigned, the adversary does not need precise vectors to estimate the (order of magnitude) of the power consumption.

the inspection, the attacker is now ready to draw the constraints necessary to design the SCT. First, the RO's dynamic power can be tweaked according to the power constraint. We remind the reader that the total power consumption can be divided into static and dynamic components as in (1). Leakage power is the static component of power and depends mainly on the threshold voltage of the transistors. On the other hand, dynamic power depends on the circuit's activity. Consequently, leakage power is proportional to the number of cells in the circuit while the dynamic power is proportional to the operating frequency. Note that, leakage power is input independent, hence, the attacker is able to acquire the original leakage assuming he/she utilizes the same corner case as the victim. Thus, to model the amplitude steps required for the RO, we need to carefully model its dynamic power consumption

$$P_{\text{total}} = P_{\text{static}} + P_{\text{dynamic}} \tag{1}$$

$$P_{\text{dynamic}} = \frac{1}{2} {V_{DD}}^2 F_{sa} \sum_{i_{\text{net}}} C_{\text{load}}(i) + F_{sa} \sum_{\text{cell}_j} E(j) \tag{2}$$

$$F_{sa} = 2F_{\text{RO}} = \frac{1}{\tau_{\text{chain}}}. \tag{3}$$

Dynamic power can be calculated using (2), where C_{load} is the capacitance load at the output nets, F_{sa} is the switching activity factor, V_{DD} is the supply voltage, and E is the total energy of a cell. The switching activity factor describes how many switches will occur per second. As for the RO, since the signals are always switching, this factor is two times the RO's oscillation frequency, which can be estimated by calculating the total path delay of the ring as in (3).

Our proposed SCT consists of only standard cells from the foundry-provided library. No custom cells or custom design techniques are involved. In addition, we utilize dedicated delay cells from the library for implementing the delay branches; these cells can provide delays in the range of 50 ps to 4 ns each. Thus, the total path delay of the RO is estimated using (4), where τ_{delay} is proportional to the number of delay cells in the active path times the delay of each cell (τ_{dcell}), see (5). The τ_{inverter} delay is from the inverter cells in the feedback path, described by (6). The τ_{control} delay is from the logic cells that controls the active paths, described by (7). Since τ_{inverter} and τ_{control} are fixed for a given implementation, τ_{delay} is the knob utilized to change the frequency of oscillation dynamically. Therefore, the RO can be designed by choosing the adequate number of delay cells in each individual branch as well as the (static) number of inverter cells in the feedback path

$$\tau_{\text{chain}} = \tau_{\text{delay}} + \tau_{\text{inverter}} + \tau_{\text{control}} \tag{4}$$

$$\tau_{\text{delay}} = (N_{D1} + N_{D2} + N_{D3} + N_{D4})\tau_{\text{dcell}} \tag{5}$$

$$\tau_{\text{inverter}} = N_i \tau_{\text{invcell}} + \tau_{\text{nand}} \tag{6}$$

$$\tau_{\text{control}} = 7\tau_{\text{and}} + 3\tau_{\text{or}}. \tag{7}$$

Finally, the equations above give a first-order estimation of the power profile of the RO.

B. Side Channel Trojan Insertion

After designing the SCT, the next step is its insertion. The attacker can utilize the ECO feature provided by commercial EDA tools for inserting the SCT. This feature can be used in two scenarios to fix small bugs in finalized layouts, pre- and post-mask. For post-mask, the ECO is used to perform slight modifications in a finalized layout after its manufacturing. A special type of logic cell, called a spare cell, is utilized to enable the post-mask ECO. Spare cells are typically inserted in commercial ICs and, when needed, are instantiated by the ECO flow. Doing so can generate a new design with minimal changes in the fabrication mask set. In the case of premask ECO, the finalized layout was not manufactured yet; thus, *it does not require any special cell*. If necessary, the ECO repurposes empty spaces to add new cells to the layout.

The main reason for utilizing the premask ECO is to avoid the time-consuming reimplementation of a design. In addition, the premask ECO is a one-time operation. Therefore, this is precisely what the attacker wishes when inserting any sort of additional logic, an automated and fast manner for modifying the target layout. Hence, our attack leverages the premask ECO for inserting the SCT.

For the SCT insertion via ECO, an attacker can achieve his/her goal without utilizing spare cells. Since we previously established that the attacker can discern any gate in a layout, the attacker can replace both filler and spare cells for his malicious logic. Contrarily to spare cells, every layout of a digital circuit has filler cells. During placement, EDA tools have to spread the standard cells to assure routability, thus mandatorily leaving gaps between cells. For more details about the relationship between placement density and HT insertion, we direct the reader to [25].

According to Fig. 2, the ECO flow is the last step for the SCT insertion. After the ECO, the attacker has to perform timing sign-off to guarantee that the performance of the victim's design was not disturbed. The SCT insertion is not likely to perturb the target's performance; it is only connected to a register (crypto key storage) and some control signals, adding a small capacitance load to them. Besides, the coupling capacitance inserted by the additional routing wires is minimal due to the SCT's lightweight characteristic and the inherent goal of the ECO flow: *not to disturb the existing logic*. However, even if unlikely, the addition of the SCT could hinder the target performance. In that case, it means that the size constraint used for designing the SCT was inappropriate. Since the ECO makes this type of attack relatively fast, the attacker can try different SCT architectures, and constraints, until it fits the target.

The attacker also has to check whether the SCT itself has timing violations. If so, the optional DV must be included to slow the SCT clock (w.r.t. the system clock). Every division by two requires one additional D-type flip-flop.

IV. Implementation and Simulation Results

In this section, we demonstrate our methodology and justify our chosen target designs, i.e., the crypto cores that we are going to insert our trojans on. The first part of the experiments

TABLE II
NAMING CONVENTION FOR THE CRYPTO CORES REGARDING THEIR FREQUENCY AND PLACEMENT DENSITY

Core	Frequency	Placement density	Acronym
AES	Low	Low	*AES_LFLD*
		High	*AES_LFHD*
	High	Low	*AES_HFLD*
		High	*AES_HFHD*
PST	Low	Low	*PST_LFLD*
		High	*PST_LFHD*
	High	Low	*PST_HFLD*
		High	*PST_HFHD*

are from simulations done using industry-grade EDA tools. After validating the methodology through simulation, the next step is the demonstration of our ASIC prototype, discussed in Section V.

A. Targets and Conditions

For our experimental investigation, we have utilized AES-128 and Present (PST) [26] crypto cores with $N_{\text{key}} = 128$ and $N_{\text{key}} = 80$, respectively. The AES crypto core was chosen due to its standardized status and popularity, while PST was chosen due to its lightweight characteristic [21].

To allow the analysis of SCT insertion for both AES and PST cores regarding changes in *frequency* and *placement density*, the combination of these variables is explored in Table II. In the column titled "Acronym," we define the terminology used for referring to the many variants of the cores. Results from a physical synthesis of the considered targets are presented in Table III. A 65-nm commercial CMOS technology was utilized to exercise very challenging placement densities (e.g., 75% for AES_LFHD) and frequencies (e.g., 0.95 GHz for PST_HFLD). The values reported are for a typical process corner (TT) and a nominal temperature of 25 °C.

B. SCT Design Results

Initially, all studied cores were physically synthesized for the placement and frequency conditions set above. These results are obtained from an industry-grade physical synthesis tool and are reported as pre-ECO results in Table III ("Before SCT insertion"). As we assume the attacker has no means to stop the clock delivery to the entire circuit, we have included the clock tree (CT) power in our results, as it has to be accounted for in the SCT power constraint. Notice how the CT power is significant compared to the targets' leakage power, even for the LF variants. Different SCTs were designed for each target by setting a power budget for the SCTs to be 10% of the sum of leakage and CT power. Since our leaking scheme does not require a high signal-to-noise ratio (SNR) to operate; we argue the 10% margin is a good trade of capability of leaking the bits and stealthiness. Importantly, this is *not* a limitation of the methodology; an attacker can pick any other threshold and still design the SCT accordingly.

With the goal of obtaining a better understanding of the static power consumption of the cores, we performed SPICE Monte Carlo (MC) simulations using a commercial circuit simulator. The MC simulation was performed for 10 000 samples,

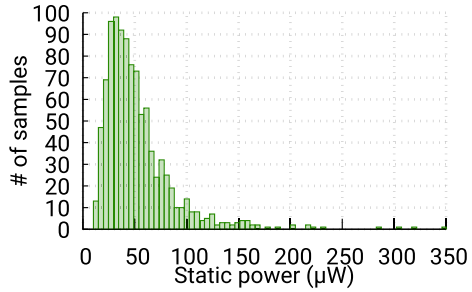

Fig. 3. PST_HFLD static power histogram, 10K MC samples (from [10]).

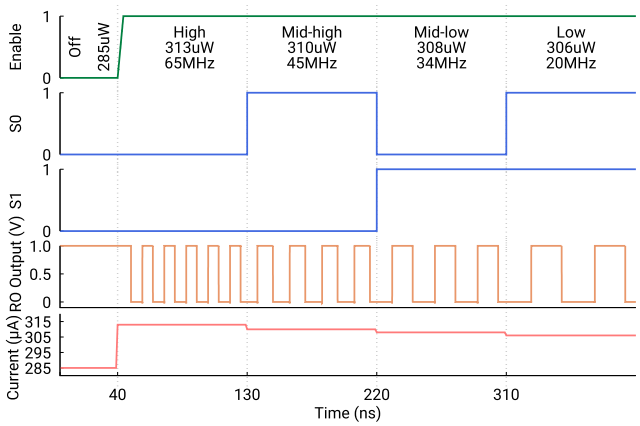

Fig. 4. Post-layout simulation of our SCT architecture in Cadence Spectre. The target design is AES_LFHD and the Trojan payload is configured as RO_{D6I10} (modified from [10]).

varying only the process characteristics, with the temperature fixed to 25 °C. Fig. 3 depicts the static power distribution of the PST_HFLD core. As expected, there are obvious outliers. Nevertheless, the distribution average matches the value reported in Table III for the typical corner. We will later show that the SCT is implemented in the very same region of the IC as the target, therefore we can also expect the same shifts in power due to process variation. This is an important observation: if a given die falls closer to the FF corner than to TT, it will be faster and more power hungry—-but so will be the trojan, nearly at the same rate. Therefore, we argue that we can safely utilize the nominal power values reported in Table III for RO design, regardless of the quality of the fabricated silicon.

Once the power constraint has been established, the attacker can proceed to estimate the multiple operating frequencies of the RO (and their associated power values that effectively leak the key). For this goal, we have taken each of our SCTs and performed SPICE simulations. The oscillation frequency and power consumption of the ROs are reported in Table IV, where each RO has been termed with a "DXIY" suffix, where X and Y represent the amount of delay and inverter cells, respectively. Notice how we do not differentiate density in the results reported in Table IV: either the trojan fits or it does not. The SCT design is nearly agnostic to placement density; this is the reason why the table contains four entries instead of eight.

To help the reader better visualize the operation of the SCT, the illustration in Fig. 4 displays an SPICE simulation of the

TABLE III
PHYSICAL SYNTHESIS RESULTS FOR OUR CONSIDERED TARGETS, BEFORE, AND AFTER TROJAN INSERTION

Core	Frequency (MHz)	Before SCT insertion Density (%)	Leakage (μW)	CT (μW)	Total Power (μW)	After SCT insertion Density (%)	Leakage (μW)	CT (μW)	Total Power (μW)	Δ Density (%)
AES_LFLD	100	61	77.4	115.2	1670	63.45	80	115.8	1720	+2.45
AES_LFHD	100	75	75.8	116.7	1660	78.20	79	117.6	1720	+3.20
AES_HFLD	1000	58	1048	1228	22800	59.37	1052	1238	23015	+1.37
AES_HFHD	1000	72	1036	1241	22610	73.02	1040	1252	22830	+1.02
PST_LFLD	95	53	14.13	32.05	371.3	67.33	20.71	34.75	483.4	+14.33
PST_LFHD	95	70	14.09	31.89	371.2	82.05	17.72	32.85	428.5	+12.05
PST_HFLD	950	52	34.02	325.30	3744	60.89	36.85	338.1	4022	+8.89
PST_HFHD	950	69	34.13	329.10	3785	80.26	36.96	341.5	4015	+11.26

TABLE IV
RO OPERATING FREQUENCY AND POWER CONSUMPTION FOR FOUR VARIANTS OF AES AND PST

Target core	RO	Power & Frequency (μW & MHz) S=00	S=01	S=10	S=11
AES_LF	RO_{D6I10}	19@65	17@45	15@34	13@20
AES_HF	RO_{D10I10}	198@551	182@483	161@390	140@300
PST_LF	RO_{D6I4}	16@112	11@58	10@39	8@20
PST_HF	RO_{D8I10}	42@79	36@61	31@46	26@31

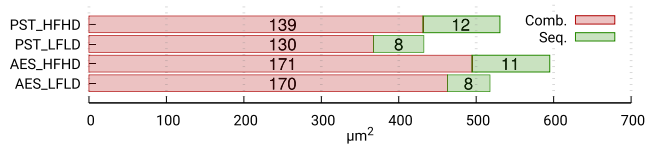

Fig. 5. Comparison of area and number of cells between SCTs (from [10]).

SCT using the AES_LFHD target as an example. The set of leaked keys in the image is {00-01-10-11}. The results for the RO are from a SPICE-level simulation with parasitics, and the total power of the AES_LFHD is estimated from physical synthesis.

We also highlight an extreme case in the RO_{D6I4} which targets the PST_LF core. For instance, both PST high-frequency versions, PST_HFLD, and PST_HFHD, are 2.55 and 2.6 times larger than their low-frequency counterparts, respectively. Here, the SCT alone represents about 10% of the size of the PST_LF core. Since area and leakage have a linear dependency, the SCT's leakage already is about 10% of the target's leakage. Hence, the power constraint is violated. However, this extreme example assumes the entire IC consists of a single PST core. For a large system-on-chip containing multiple cores, the power budget for designing the SCT would be much more forgiving.

Alongside the custom-simulated ROs, the SCTs are synthesized for each N_{key} and at the same clock frequency as of the target. Exclusively for the HF targets, we added the DV block to ensure the synchronous portion of the SCT does not violate timing. For AES_HF, the system clock was divided by eight while for PST_HF it was divided by 16. The characteristic of the SCTs is illustrated in Fig. 5, where we show the area and number of cells for each SCT.

C. SCT Insertion Results

After designing the RO and synthesizing the remainder of the SCT logic, the attacker is ready to perform the insertion via ECO. Here, *we assume the role of the attacker* and utilize the post-route layout for extracting the gate-level netlist from the targets and performing the insertion methodology described in Fig. 2. In our experiments, we complete the ECO flow in a single run, i.e., calling the ECO command a single time.

TABLE V
ROUTING LENGTH PER METAL FOR THE PST_HFHD IMPLEMENTATION, PRE- AND POST-ECO

Metal layer	Wirelength (μm) pre-ECO	post-ECO
M2	5568	5759
M3	7036	8332
M4	4580	6223
M5	3182	6417
M6	2528	5283
M7	-	706

The results for SCT insertion are described on the right side of Table III ("After SCT insertion"). For all considered scenarios, the ECO flow was capable of placing and routing the SCT successfully, even for extremely dense layouts. Considering that high cell density implies fewer routing resources, we verified that the ECO flow purposefully utilizes the least congested metal layers. This trend is noticeable in Table V,[4] where the routing length per metal layer is reported for the PST_HFHD target. Notice the significant increase in metals M5, M6, and M7. Also, note that the lower metal layers are more closely associated with the circuit performance [27], so overheads in M5 and above are unlikely to affect critical paths.

We also provide a visual comparison of the density increase for the AES_HFHD and PST_HFHD SCTs in the bottom part of Fig. 6. Note that the placement of the targets (top part of Fig. 6) was kept identical and only filler cells were removed for the SCT insertion via ECO. This is a key finding of our work and confirms the feasibility of the attack.

Besides being able to insert the SCT, the ECO flow also has to preserve the performance of the target circuit. The additional malicious logic increases the load on the paths to which the SCT is connected, and, in general, the SCT routing could increase the coupling capacitance to adjacent paths. Thus, the impact on the target performance due to the SCT insertion is related to the number of connections between them and the increase in density. For the AES implementations, the addition

[4]In our considered metal stack, M1 cannot be used for signal routing. For this reason, M1 is not shown. Similarly, M8/M9 are reserved for power distribution.

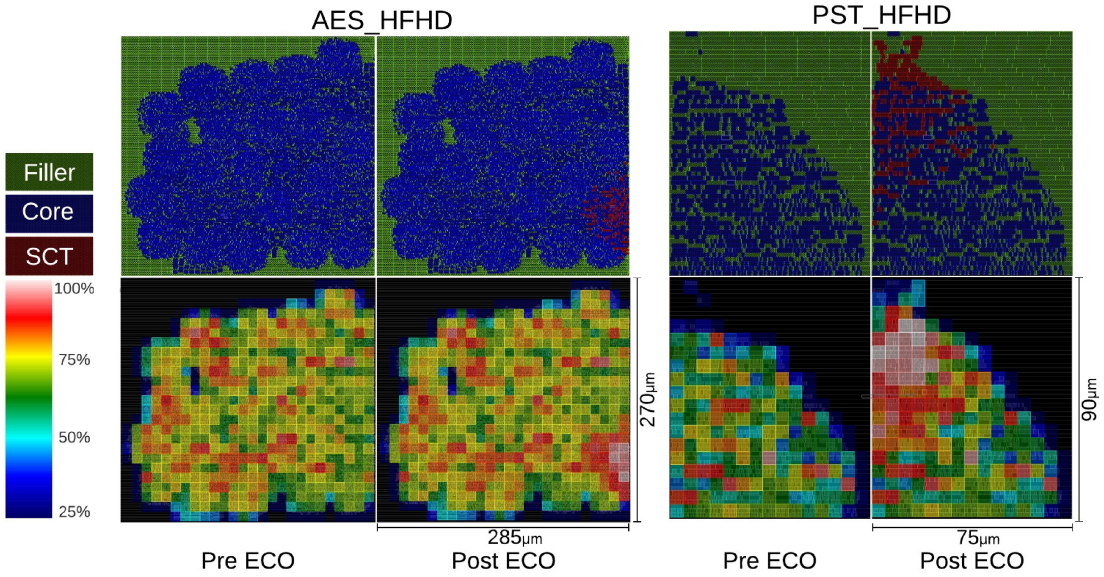

Fig. 6. Placement view (top panels) and density map (bottom panels) of the AES_HFHD and PST_HFHD cores, before and after SCT insertion via ECO (modified from [10]).

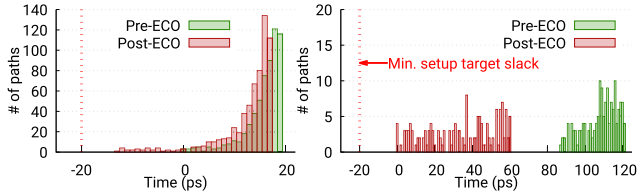

Fig. 7. Pre- and post-ECO setup timing slack comparison of AES_HFHD (right) and PST_HFHD (left) (from [10]).

of the SCT increased their total density by a small margin. On the other hand, for the PST cores, the SCT represents a large portion of the total circuit area. This is illustrated in the bottom part of Fig. 6, where the density map of the PST_HFHD and AES_HFHD layouts are shown.

The impact on the performance of the AES_HFHD and PST_HFHD cores is illustrated in Fig. 7, where we contrast the pre- and post-ECO timing slack. These results show that the impact is greater on the PST_HFHD implementation, which is explained by the high-density increase reported in Table III. One can appreciate how the red bars in Fig. 7 are shifted to the left (w.r.t. the green bars). However, this shift was not sufficient to degrade the performance of the cores. In particular for the PST core, the ECO engine completed successfully by using some of the safety margin (20 ps) we applied to all our paths. This safety margin is small and compatible with industry practices, where often margins in the range of tens of picoseconds are utilized. Thus, in terms of performance, our attack appears to be adequate for realistic commercial designs.

Furthermore, considering that the reported timing slack results are for implementations with a challenging operating frequency, we argue that our proposed methodology is not only capable of inserting an SCT in a high-density target, but also of keeping the target performance as close as possible to the original, regardless of the frequency of operation. Thus, for an attacker, inserting malicious logic by repurposing filler/spare cells with the help of an ECO feature is more than adequate, it is almost ideal. First, this methodology has an area overhead of 0% because we utilize space that is otherwise unused. Therefore, a slight increase in density is the only measurable "overhead." Second, the performance of the target is very likely to remain the same. Third, the runtime is only a fraction of other methods for inserting malicious logic, such as reimplementation. In the following section, we discuss these ECO characteristics in detail. For further details of the capabilities of the ECO and a rich discussion of insertion success, we direct the reader to [28].

V. Test Chip Design and Validation

In this section, we present our fabricated ASIC prototype and its many details. Initially, we present the chip architecture and its functionality.

A. ASIC Prototype Architecture

Our main goal while designing a silicon proof of concept for our methodology is to demonstrate the malicious potential of the ECO flow. For this purpose, we developed a full framework for performing a fabrication-type attack (see Fig. 2). Our proposed SCTs are carefully crafted in order to stress test the ECO flow and its limitations: the chosen circuits are synthesized for their maximum frequency and utilizing challenges densities, making the SCT insertion even more challenging. Our framework includes all steps necessary for assessing the GDSII database, designing an HT, and inserting it in a finalized layout.

As discussed in the previous section, our targets are AES and PST crypto cores. For our ASIC prototype, we chose 4 of the 8 versions of the crypto cores described in Section IV. These versions are the highest density ones (AES_LFHD, AES_HFHD, PST_LFHD, and PST_HDHD), purposefully

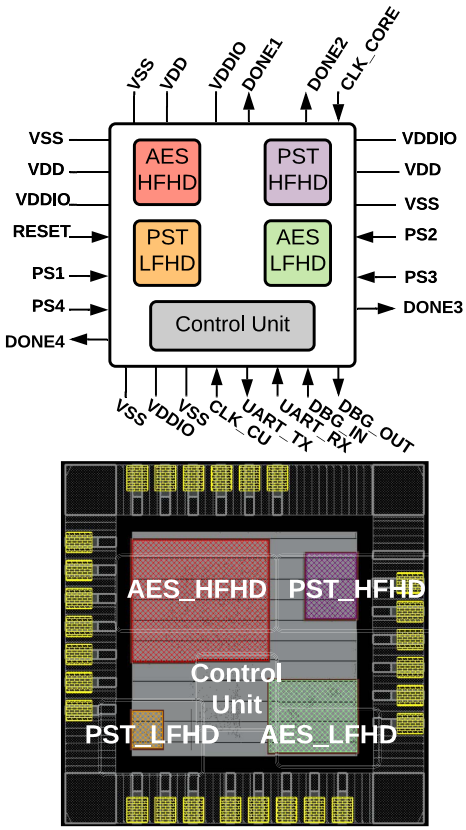

Fig. 8. Top level diagram (top panel) and floorplan of our test chip (bottom panel).

selected for the difficulty in manipulating a dense layout. The top-level architecture and the floorplan of our ASIC design are depicted in Fig. 8.

Our chip contains the four chosen crypto cores and a control unit for handling the data traffic in and out of the chip. This control unit has a UART-like communication protocol and a register bank to store the plaintext, the cryptokey, and the ciphertext. We note that the plaintext and the cryptokey can be programmed externally via UART. For communicating with the control unit, the signals UART_TX, and UART_RX are utilized. The signals DONE_1, DONE_2, DONE_3, and, DONE_4 indicate the end of a cryptographic operation for AES_HFHD, AES_LFHD, PST_HFHD, and PST_LFHD, respectively. These signals are exposed as primary outputs only for debug reasons, their presence is not required for the attack. Internally, these same signals are the triggers for the SCTs.

Our architecture has five clocks domains: the "always-on" clock is delivered by the signal CLK_CU, and the other four domains are connected to the signal CLK_CORE (which assumes four different frequencies). For switching the cores on/off selectively, the signals PS1, PS2, PS3, and, PS4 are utilized as described in Table VI. Both signals DBG_IN and DBG_OUT are used for debugging purposes only. The power-ground scheme has two power sources (VDD and VDDIO) and a common ground (VSS): VDD supplies the core cells with a nominal voltage of 1.0 V and VDDIO supplies the IO cells with a nominal voltage of 3.0 V.

TABLE VI
POWER DOMAINS, CLOCK, AVERAGE TOTAL POWER, AND LEAKAGE ACROSS THE SAMPLES TESTED

Block	Clock	Switch Signal	Leakage (μW)	Total Power (μW)
Control Unit	CLK_CU @1MHz	Always on	46.69±4.75	-
AES_HFHD	CLK_CORE @1GHz	PS1	743.79±108.07	101160±10781
AES_LFHD	CLK_CORE @100MHz	PS2	131.57±10.35	3139.32±85.38
PST_HFHD	CLK_CORE @950MHz	PS3	80.75±7.82	9661.3±758.52
PST_LFHD	CLK_CORE @95MHz	PS4	74.35±6.84	868.56±57.90

For manufacturing the chip, we have utilized a commercial 65 nm technology (the exact same technology utilized in the previous section). We also made use of three standard cell flavors (LVT, SVT, and HVT) and power switch IP for isolating power domains. Our idea in utilizing multiple voltage threshold cells is to bring our implementations in line with industry practices, thus adding another degree of realism to our attack. The power switches were utilized to create a power domain for each crypto core, making it possible to enable one core at a time on the same chip. Implementing the crypto cores with the possibility of total shut down is extremely valuable for evaluating our attack, because we are only reading the power that come from the enabled core. However, in our chip, the control unit is on an "always-on" domain, thus, this portion of the circuit is always enabled. Nonetheless, this characteristic later did not affect our tests or measurements in a negative manner. The power domain information and the related switch signals are described in Table VI.

The ideal ROs designed in the previous section (see Table IV) only consider the leakage from the crypto core alone. In our ASIC prototype, we have extra components that compete with the leakage of the currently enabled core—even if here we assume that only one core is on at a time. Therefore, the ROs require small adjustments to accommodate the extra competing leakage, which is a trivial exercise: an attacker can create a database of SCT architectures for known targets and apply small shifts to them by modulating the number of inverters or delay cells in the RO. The newly adjusted ROs for the ASIC prototype are described in Table VII. These results are from detailed SPICE-level simulations.

Our chip was designed in November 2020, fabricated at a partner foundry in March 2021, and bench tests were started in July 2021. Our bare die and its layout are contrasted in Fig. 9. For the validation of the design, we have 25 packaged samples of the chip. All packaged samples were confirmed to be 100% functional.

B. SCT Insertion

In our framework, for fully inserting the SCT into a layout, the attacker has to inspect the layout, extract the netlist, estimate the operating frequency, estimate the power consumption, modify the netlist, and finally, insert the SCT utilizing the ECO flow. The time necessary to inspect the layout in

TABLE VII
RO OPERATING FREQUENCY AND POWER CONSUMPTION FOR EACH CRYPTO CORE OF THE ASIC PROTOTYPE

Target core	RO	Power & RO Frequency (μW & MHz) S=00	S=01	S=10	S=11
AES_LFHD	RO_{D8I14}	32@90	27@61	23@46	20@31
AES_HFHD	RO_{D12I14}	249@551	227@483	198@390	169@300
PST_LFHD	RO_{D8I6}	22@169	19@90	16@46	13@21
PST_HFHD	RO_{D10I10}	30@90	24@60	20@37	17@19

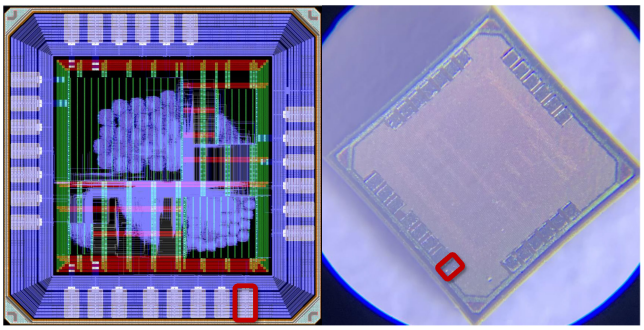

Fig. 9. Our bare die (right) and its layout (left). The lower-right corner is identified by the highlighted pin.

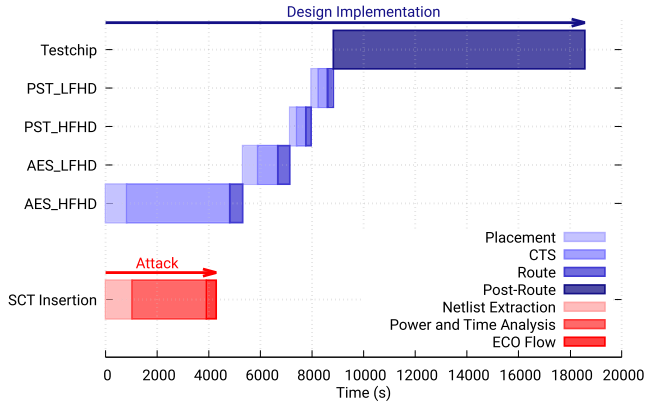

Fig. 10. Physical implementation execution time (s) for each step of the flow, and, execution time (s) for inserting the SCT in each implemented crypto core.

order to find the security-critical nodes depends on the expertise of the attacker, which makes it very difficult to estimate. On the other hand, the other steps can have their runtime measured. These times are depicted in red in Fig. 10 and are contrasted with the total time required for the physical implementation of the original design, shown in blue. In our case, the required time for implementing the original cores and building the top-level layout is about 5 h and 9 min. The most time-intensive tasks are the clock-tree synthesis and the post-route optimizations. For our test chip, the post-route step also includes the assembling of the cores, the top-level clock-tree synthesis, the top-level routing, and a chip-level design rule check. The netlist extraction was executed in a server with an *Intel Xeon* 5122 CPU @ 3.6 GHz and 96 GB of RAM, while all other tasks were executed in a server with an *Intel Xeon* X5690 @ 3.47 GHz and 768 GB of RAM. When multithreading was supported for a given task, the number of concurrent CPUs in use was set to 8. All execution times were measured five times and the presented values are the average execution time.

For the attack, the first task is extracting the netlist from the layout, which is done in 17 min. Then, the estimation of the operating frequency and power consumption is done in 48 min, which includes full parasitic extraction, static timing analysis, and a power analysis—all utilizing the highest effort available. The static timing analysis was done utilizing two corners and the power analysis only one. Differently of a chip sign-off task executed by a designer, where all corners are considered, the attacker can make use of a couple of corners only for a representative evaluation of the IC operation. Inserting the SCT is done individually for each core, and the total time for inserting all four SCTs is only 6 min. Thus, the entire attack for this design requires 1 h and 11 min. Had an attacker decided to modify the layout by reimplementing the extracted netlist, the total time of the attacks jumps to 6 h and 21 min. Therefore, by using an ECO flow, not only the original design remains untouched—i.e., the cells' placement and routing remain the same after the attack—the attacker can drastically reduce the attack time. This is true for both SCTs and functional trojans.

For large ICs, how can an adversary insert an HT in a limited time frame? Or, in other words, how does the attack time scale with the complexity of the IC? To answer this question, we ought to look at the individual steps of the attack. First, for the netlist extraction, there are no alternative routes for avoiding/bypassing this specific task in the EDA tools. However, the execution time scales fairly with the size of the circuit.

On the other hand, the timing and power analysis can become overwhelming depending on the circuit size. Fortunately for the attacker, the runtime for these tasks can be reduced by utilizing different effort levels and/or by reducing the number of corners utilized. Another option is utilizing a wire-load model instead of a more precise RC model for the parasitic extraction. All of those alternatives would change the quality of the analysis, e.g., the static timing analysis in low effort with only the best-case corner will report a very optimistic timing requirement for setup (i.e., a higher operating frequency). From the attacker's point of view, an optimistic timing analysis would lead to a higher power budget, making the SCT less stealthy but more likely to succeed. An overall picture of the tradeoffs between the analysis quality, the SCT insertion runtime, SCT stealthiness, and the attack success rate is provided in Table VIII. Here, we clarify that the number of corners considered for the timing analysis is always two (BC/WC analysis). For the power analysis, only one corner is taken into account. The RC model utilized for the low effort is the wire-load model; other efforts utilize more precise models.

It must also be highlighted that the ECO insertion time does not scale with chip size: the ecoPlace and ecoRoute engines are available within a physical synthesis tool specifically for the purpose of performing local changes.

Thus, even for large ICs, the attacker can significantly reduce the time required for inserting the SCT if he/she is willing to compromise the trojan stealthiness or its success rate. But, most importantly, the time required for SCT insertion

TABLE VIII
TRADEOFF COMPARISON BETWEEN THE SCT INSERTION RUNTIME, SCT STEALTHINESS, AND THE ATTACK SUCCESS RATE, FOR DIFFERENT EFFORT CONFIGURATIONS OF PARASITIC EXTRACTION, STATIC TIMING ANALYSIS, AND POWER ANALYSIS

Cut Layout	Parasitic Extraction	Static Timing Analysis	Power Analysis	Runtime	SCT Stealthiness	Attack Success Rate
No	Low	Low	Low	Short	Weak	Medium
No	Medium	Medium	Medium	Average	Strong	Medium
No	High	High	High	Very Long	Very Strong	Very High
Yes	High	High	High	Short	Very Strong	Very High

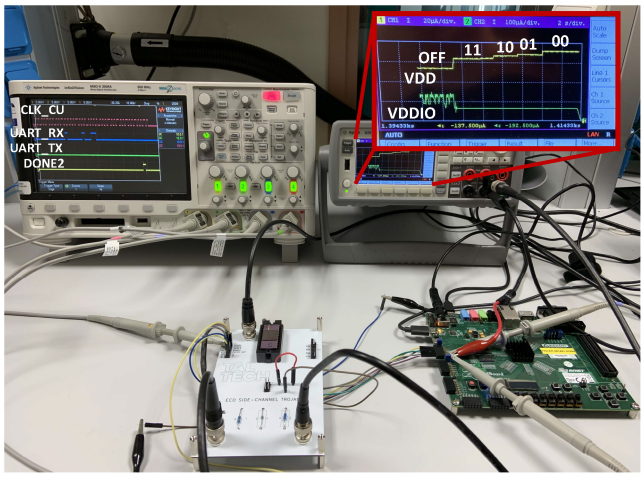

Fig. 11. Setup used for bringing up the test chip. On the left side, we show the signals used for controlling the chips. On the right side, the current consumption of the chip when the RO is active.

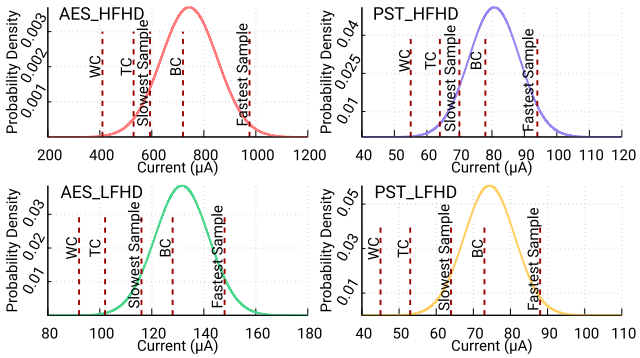

Fig. 12. Leakage distribution for each crypto core contrasted with the leakage from physical synthesis report for three corner cases, and the leakage of outlier samples.

when utilizing our proposed framework is always a fraction of the time required for implementing the original design (comparatively shown in Fig. 10). This relationship does not depend entirely on the size of the circuit.

As shown on the last row of Table VIII, an adversary could potentially make a "cut" in the GDSII database and treat only a part of it. To some extent, this would be the equivalent of doing a block analysis instead of a schip analysis. The possibility of cutting a layout, although supported by the tools, is not studied in this work for the reason that an adversary would have to execute said cut very precisely and, after the SCT insertion, merge the modified and original layout, a task that is not trivial and could potentially damage the original layout if mishandled.

C. Tests and Measurements

For the purpose of bench testing our ASIC prototype, we have designed a custom printed circuit board (PCB). The PCB itself only contains passive components utilized for helping with the measurements and filtering noise from the power supplies. An attacker does not need our PCB and/or test setup to mount the attack. The chip is controlled by a ZedBoard from Avnet with a Xilinx Zynq-7000 All Programmable SoC. Our complete bench test setup is shown in Fig. 11, where we also make use of a 4-channel digital oscilloscope and a 2-channel power supply with an ammeter with pico ampere precision.

The first phase of the validation was to measure total power and leakage power from each block across all 25 samples. For this, all the primary inputs were set to "0," and each core was enabled one at a time by asserting its respective switch signal (see Table VI). For the total power, we delivered the clock signal for each block utilizing the specified operating frequency. The results of the total power average and leakage are given in Table VI, and its distribution across the samples is depicted in Fig. 12. These results are in line with the expected from the power reports done during the physical implementation, as these results are contrasted in Fig. 12 for the worst, typical, and, best-case scenarios (SS-0.9 V-0 °C, TT-1 V-25 °C, FF-1.1 V-125 °C, respectively). The corners provided by the vendor are for extreme cases, i.e., the best-case scenario is characterized at 125° with an over voltage of 1.1 V, in our case the measurements were done at room temperature and at a nominal voltage of 1.0 V. Our samples are skewed toward the best-case scenario, demonstrating higher average leakage. The slowest sample is near the typical case while the fastest sample is very far from the expected best case. Thus, our samples have a high variance between them (i.e., their leakage values are very spread from the mean), with variance values of 1212, 102, 59, and 44 for the AES_HFHD, AES_LFHD, PST_HFHD, and PST_LFHD cores, respectively.

In the second phase of the experiments, we assessed the SCTs and the feasibility of the attack. This was done by the following procedure.

1) A cryptokey with the 8 first bits set to "11-10-01-00" was programmed in the Control Unit's register bank.
2) A command for a single encryption was issued.
3) Right after the encryption is done, the chip asserts one of the "DONE" outputs to mark the time at which the RO starts operating.
4) Using only the clock signal CLK_CORE, three bursts of clocks were sent in order to shift the cryptokey connected to the RO three times.
5) During the whole procedure, the current consumed by the chip is monitored.

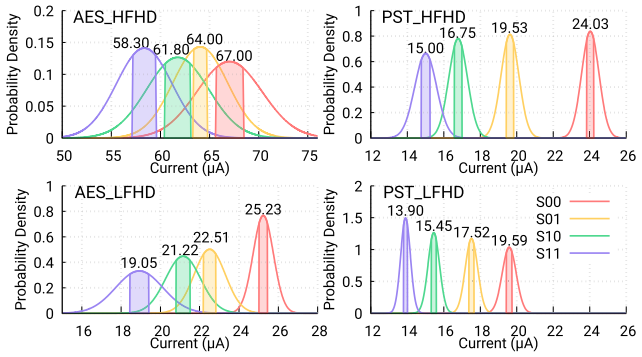

Fig. 13. Power consumption "steps" distribution for each crypto core. The shadowed area represents the 95% confidence interval.

An example of this procedure for the AES_LFHD core is shown in Fig. 11; the "UART_TX" signal carries the single encrypt command. As a visual aid, as soon as the "DONE" signal is asserted, the clock sources are turned off in this example (in Fig. 11 only CLK_CU is shown). As clearly depicted in the ammeter, there are discrete steps representing the leaked bits "11-10-01-00" from left to the right, respectively, as expected from the key programmed for this experiment. This was repeated three times for each core of each chip to confirm the behavior. The measured current values were approximated to normal distributions, as represented in Fig. 13.

By comparing the RO performance from the simulations (see Table VII) with ASIC measurements, it is clear that the slowest ROs are performing as expected. However, the fastest RO targeting the AES_HFHD core can only operate at a low frequency, generating a power step of about 25% of what was expected. In this case, the ECO insertion had to spread the RO cells farther away because of the lack of empty spaces nearby (see Fig. 6). For this core, the planned power steps were in the order of 200 μA, and the actual power steps after manufacturing were in the order of 60 μA. However, the attack will still enjoy a high chance of success due to the distinct separation of the power steps, even if 95% confidence intervals of the distributions almost overlap. We have also confirmed that the interconnect delay wire load was higher than we expected from SPICE-level simulations. For extreme cases such as the AES_HFHD core, the attacker has to make extra considerations for implementing an RO with high power consumption, in case he/she desires high fidelity from the RO power steps. The best-case scenario is achieved for PST_LFHD (see Fig. 13, bottom right): there is absolutely no overlapping between adjacent steps, with very low variance, which highly increases the success rate of the attack.

The experimental measurement results obtained show that the variability in the manufacturing process does not affect the effectiveness of the RO for the smaller designs (AES_LFHD, PST_LFHD, and PST_HFHD), meaning that the attack can be carried out with the same probability of success, regardless of the silicon quality for a given sample. The quality of the silicon impact on the SCT performance is directly connected with its size and how the cells were placed. If the cells are heavily spread, as occurred in the AES_HFHD, the variance of the steps is very high when compared with an SCT with a similar size (see Fig. 5) placed with low spread, as in the PST_HFHD. We hypothesize that the higher the physical spread between the cells that compose the SCT, the more susceptible to local variation they become. Visually, this can be seen by the width of the shadowed areas in Fig. 5.

However, we learned from our experimental results that even in a high-spread scenario, the attack is successful for all implemented cores. For a single given die, there is no difficulty in differentiating the four possible leakage states associated with the two bits of the key. Moreover, these results make it evident that our SCT is successfully capable of creating distinct steps with a 2-μA precision. Furthermore, the induced power consumption for the smaller crypto cores (PST_LFHD) was in the 20-μW range. This makes our SCT a perfect fit for targeting designs that consume very low power while maintaining a reasonable level of stealthiness.

VI. Discussion

For an SCT insertion framework like ours, its effectiveness can be determined by three characteristics: 1) the success rate of the attack; 2) probability of detection (i.e., its stealthiness); and 3) feasibility of the insertion of the malicious logic during the fabrication-time attack. As we have already discussed, our SCT was successful in (1) by making the attack of leaking the cryptokey viable, i.e., the attack was fully accomplished. Nevertheless, we have not yet discussed the probability of our SCT being detected. Detecting a trojan of any kind is generally a difficult task [29]. For SCTs, any method that relies on observing corrupted bits or any degree of incorrect computation is bound to fail—SCTs do not alter the functionality of the device under attack.

Because of the inherent opaqueness of ICs, inspecting their internal components is not trivial. Therefore, methods for inspecting ICs are separated into two classes, invasive and noninvasive. As the name suggests, invasive procedures require access to the IC's internal parts, which may destroy the inspected sample. Contrarily, noninvasive methods analyze the IC from the outside.

An IC internal structure can be reconstructed utilizing advanced physical reverse engineering techniques [30], [31], [32]. Physical reverse engineering is often an invasive method divided into three steps: 1) sample preparation; 2) delayering and imaging; and 3) image post-processing. The chip must first be depacked by wet-chemical or mechanical means to access the die. Then, after recovering the bare die, the IC has to be delayered, and each layer has to be optically captured using a scanning electron microscope (SEM) or focused ion beam (FIB). Finally, the images are stitched and vectorized to retrieve the layout representation of the chip [31]. Note this process is yet to be fully automated, resulting in a highly time-intensive task prone to errors [30]. On the other hand, promising methods are utilizing x-ray diffraction or a confocal microscope to recover the inner structure of the IC without the need for delayering [32], i.e., without destroying the inspected sample. Yet, these methods require access to costly and not rapidly available equipment. Nonetheless, our SCT is likely to be detected by any of these methods due to its size and

amount of connected wires. We emphasize that those physical reverse engineering methods are not a standard practice of the IC industry to perform this analysis.

Differently, noninvasive methods include analyzing the physical characteristics of the IC, and/or, the behavior of the IO signals (i.e., timing and state) [11]. Our SCT does not disrupt any data path and our insertion methodology also does not interfere with the overall performance of the target. Thus, the probability of it being detected by analyzing the IO signals is effectively zero. Detection techniques that consider the path delay, e.g., path delay fingerprint [33], would have a low probability to detect our SCT. Nonetheless, our SCT changes the overall power consumption of the target. First, the extra leakage could raise a red flag if the IC is thoroughly inspected. The chance of being detected in this type of inspection is related with the percentage of the extra leakage from the SCT. This is also true for any HT that inserts additional logic. However, if the percentage is insignificant compared with the target, the extra leakage has a high chance to be interpreted as a skew from the manufacturing process and/or imprecision of the measurements. Second, the artificial extra consumption when the SCT is triggered can also be a red flag. In this scenario, the engineer conducting the inspection would need to know the exact moment when the SCT is triggered to suspect any alteration. Specialized detection methodologies have been proposed that utilize leakage and total power as input [34], [35]. By utilizing these advanced methods of detection, our SCT could be detected due to the trigger scheme. Since our SCT is triggered after each cryptographic operation, a periodic power fluctuation would be visible. However, the trigger scheme utilized in our silicon validation can be further modulated by an attacker by creating rarer trigger conditions, making the SCT stealthier.

In our threat model, we assumed the attacker only has access to the layout and utilizes the extracted netlist for inserting the SCT. This netlist does not contain any node names, making it impossible to distinguish nodes of interest by name. Thus, the attacker has to identify the circuit functionality by inspecting the layout for "clues." In the case of AES, this target circuit can be easily identified in a layout because of its implementation regularity, and, from there, the same holds true for the nets/registers that carry the cryptokey. On the other hand, visually locating a Present core (or any other core) would be a more challenging task. Nevertheless, visual inspection is not the only technique that can be utilized for searching for security-relevant nodes. Based on the ECO framework herein presented, Hepp et al. [28] created a framework for blindly inserting HTs. The results presented in [28] further demonstrated the ECO capabilities for inserting HTs. On top of that, their framework does not require additional knowledge besides the victim's layout for locating security-relevant nodes.

A clear advantage of our SCT architecture is its robustness to manufacturing process skew. As demonstrated in the previous section, even with a large difference in performance between the fastest and slowest sample (see Fig. 12), the SCA was successful (see Fig. 13). When compared with a similar approach presented in [16], our SCT architecture does not need any workaround to be implemented because of the performance of the technology. Even more, our flow anticipates difficulties during the SCT insertion by including the optional DV. However, our architecture has the disadvantage of being large in comparison with other similar SCTs. The size of our SCT is proportional to the number of bits leaked at a time and to the total number of bits intended to be leaked (i.e., the SCT is proportional to the key length). This characteristic increases the probability of detection.

Our attack is arguably prevented by a few techniques. Split Manufacturing [4], as mentioned before, is a powerful prevention technique for HT insertion overall, where the attacker has access only to the layer that contains the devices—the connections between them are hidden from the untrusted foundry. Hence, the attacker would only be able to find the nodes by visual inspection without any connection information, making the SCT insertion a "blind" effort. Another relevant technique is the insertion of dummy cells and routing wires [36], [37] with the intent to reduce the empty spaces where—potentially—an HT could be inserted. As demonstrated in this work, our insertion methodology overcomes incredibly high densities. Hence, these techniques would only be effective if the entire chip is populated with dummy cells and routing wires, increasing the design density above 95%, which for new technologies is very challenging and can potentially hinder the IC performance. On top of that, dummy cells have transistors inside them, which increases the leakage of the chip proportionally to the number of additional dummy cells. Thus, the leakage overhead could be in the range of 40%–50%—assuming that a typical design has an approximate density in the range of 60%–50%. For industrial designs, this type of technique might not be practical in terms of the potential performance loss, making the tradeoff between security and power consumption not interesting for many vendors and applications. Therefore, the adoption of these techniques as a countermeasure against malicious logic is very unlikely.

Another metric to qualify an SCT insertion attack is the total time required to perform the attack. Our threat model assumes the attack occurs in the untrusted foundry and only the layout is accessible to a rogue element. Foundries are typically working at full capacity year round, hence, the timing window that the layout is processed to begin the manufacturing is limited. This time window is precisely the period of time in which a rogue element has to mount a fabrication-time attack. In recent SCT works that contemplate silicon validation [15], [16], [21], the techniques for inserting the malicious logic are not disclosed—making it difficult to address if the attack can be replicated in a realistic scenario.

Placing and routing an SCT manually is a time-intensive task and prone to errors, even if the HT design has only a dozen of cells. Thus, the insertion of an SCT has to be automated by utilizing an EDA tool. Inserting an SCT by reimplementing the design has a significant runtime, in the case of our test chip (see Fig. 10) it is required at least 7 h and 18 min. Replicating an entire chip without the original timing and power constraints could be very difficult, which can potentially affect the target performance, thus decreasing the stealthiness of the attack. In the case of the ECO flow, the runtime for inserting the SCT in our test chip is

only 1 h and 11 min. Nonetheless, as previously alluded, the ECO flow has the advantage of keeping the original design untouched which increases the stealthiness of the attack. Even more, our proposed ECO flow does not require the original power and timing constraints, an estimation can be used (see Section III-A). Consequently, our proposed ECO flow method for inserting not only SCTs, but any type of malicious logic, is arguably a superior option. Furthermore, the short runtime associated with the ECO flow makes the fabrication-time attack feasible in a realistic scenario, where the time window that a rogue engineer has for modifying the layout is (very) limited.

VII. Conclusion

In order to steal secret information from crypto-capable ICs, a rogue element within the foundry may insert an SCT. The SCT architecture described in this work has the advantage of not violating any design specification of the target circuit, nor is any datapath obstructed. This is all possible because of the use of an ECO flow for inserting SCTs. Since this feature is readily available, adversaries may maliciously utilize it.

Our findings and results from the validation of our ASIC prototype demonstrated the feasibility of the framework—from the layout inspection to the actual attack. The attack was successful for all 25 samples available, successfully extracting the cryptokey via power signatures. The measurements have also demonstrated the robustness of the SCT against skews from the manufacturing process.

For our test chip, all four SCTs were inserted in less than two hours. Consequently, the attack would be viable in a real fabrication-time attack. As a venue for future work, we intend to improve the search of security-relevant nodes for inserting HTs in order to understand if a capable adversary is able to execute a "blind" yet successful insertion. This line of work has already been pursued in [28], but we believe that, when armed with an ECO-based insertion engine, there are still many other ways in which a clever adversary can succeed.

References

[1] M. Lapedus. "Big trouble at 3nm." 2018. [Online]. Available: https://semiengineering.com/big-trouble-at-3nm/

[2] 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," *Proc. IEEE*, vol. 102, no. 8, pp. 1207–1228, Aug. 2014.

[3] S. M. Ben. "Security challenges and requirements for industrial control systems in the semiconductor manufacturing sector." 2012. [Online]. Available: https://csrc.nist.gov/CSRC/media/Presentations/Security-Challenges-and-Requirements-for-Control-S/images-media/presentation-3_salem.pdf

[4] T. D. Perez and S. Pagliarini, "A survey on split manufacturing: Attacks, defenses, and challenges," *IEEE Access*, vol. 8, pp. 184013–184035, 2020.

[5] K. Zamiri Azar, H. Mardani Kamali, H. Homayoun, and A. Sasan, "Threats on logic locking: A decade later," in *Proc. GLSVLSI*, 2019, pp. 471–476.

[6] M. Yasin, J. Rajendran, and O. Sinanoglu, *Trustworthy Hardware Design: Combinational Logic Locking Techniques*. Cham, Switzerland: Springer Publ. Company, Incorp., 2019.

[7] A. Chakraborty et al., "Keynote: A disquisition on logic locking," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 10, pp. 1952–1972, Oct. 2020.

[8] M. Li et al., "Provably secure camouflaging strategy for IC protection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 8, pp. 1399–1412, Aug. 2019.

[9] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *Proc. IEEE HOST*, 2015, pp. 137–143.

[10] T. Perez, M. Imran, P. Vaz, and S. Pagliarini, "Side-channel trojan insertion—A practical foundry-side attack via ECO," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2021, pp. 1–5.

[11] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design Test. Comput.*, vol. 27, no. 1, pp. 10–25, Jan./Feb. 2010.

[12] M. Rostami, F. Koushanfar, and R. Karri, "A primer on hardware security: Models, methods, and metrics," *Proc. IEEE*, vol. 102, no. 8, pp. 1283–1295, Aug. 2014.

[13] L. Lin, W. Burleson, and C. Paar, "MOLES: Malicious off-chip leakage enabled by side-channels," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2009, pp. 117–122.

[14] L. Lin, M. Kasper, T. Güneysu, C. Paar, and W. Burleson, "Trojan side-channels: Lightweight hardware trojans through side-channel engineering," in *Cryptographic Hardware and Embedded Systems*. Heidelberg, Germany: Springer, 2009, pp. 382–395.

[15] Y. Jin and Y. Makris, "Hardware trojans in wireless cryptographic ICs," *IEEE Design Test Comput.*, vol. 27, no. 1, pp. 26–35, Jan./Feb. 2010.

[16] Y. Liu, Y. Jin, and Y. Makris, "Hardware trojans in wireless cryptographic ICs: Silicon demonstration & detection method evaluation," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, 2013, pp. 399–404.

[17] R. Kumar, P. Jovanovic, W. Burleson, and I. Polian, "Parametric trojans for fault-injection attacks on cryptographic hardware," in *Proc. Workshop Fault Diagnosis Tolerance Cryptogr.*, 2014, pp. 18–28.

[18] K. Yang, M. Hicks, Q. Dong, T. Austin, and D. Sylvester, "A2: Analog malicious hardware," in *Proc. IEEE Symp. Security Privacy (SP)*, 2016, pp. 18–37.

[19] J.-F. Gallais et al., "Hardware trojans for inducing or amplifying side-channel leakage of cryptographic software," in *Trusted Systems*, L. Chen and M. Yung, Eds. Berlin, Germany: Springer, 2011, pp. 253–270.

[20] L. Ali and Farshad, "Analog hardware trojan design and detection in OFDM based wireless cryptographic ICs," *PLoS One*, vol. 16, no. 7, 2021, Art. no. e0254903. [Online]. Available: http://dx.doi.org/10.1371/journal.pone.0254903

[21] S. Ghandali, T. Moos, A. Moradi, and C. Paar, "Side-channel hardware trojan for provably-secure SCA-protected implementations," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 6, pp. 1435–1448, Jun. 2020.

[22] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology (CRYPTO)*, M. Wiener, Ed. Berlin, Germany: Springer, 1999, pp. 388–397.

[23] R. S. Rajarathnam, Y. Lin, Y. Jin, and D. Z. Pan, "ReGDS: A reverse engineering framework from GDSII to gate-level Netlist," in *Proc. IEEE Int. Symp. Hardw. Oriented Security Trust (HOST)*, 2020, pp. 154–163.

[24] A. Jain, Z. Zhou, and U. Guin, "Survey of recent developments for hardware trojan detection," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2021, pp. 1–5.

[25] T. Trippel, K. G. Shin, K. B. Bush, and M. Hicks, "ICAS: An extensible framework for estimating the susceptibility of IC layouts to additive trojans," in *Proc. IEEE Symp. Security Privacy (SP)*, 2020, pp. 1078–1095. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP40000.2020.00083

[26] A. Bogdanov et al., "PRESENT: An ultra-lightweight block cipher," in *Cryptographic Hardware and Embedded Systems*, P. Paillier and I. Verbauwhede, Eds. Heidelberg, Germany: Springer, 2007, pp. 450–466.
%bibitembib27

[27] S. N. Pagliarini, M. M. Isgenc, M. G. A. Martins, and L. Pileggi, "Application and product-volume-specific customization of BEOL metal pitch," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 9, pp. 1627–1636, Sep. 2018.

[28] A. Hepp, T. Perez, S. Pagliarini, and G. Sigl, "A pragmatic methodology for blind hardware trojan insertion in finalized layouts," in *Proc. Int. Conf. Comput.-Aided Design*, 2022, p. 9.

[29] S. Bhasin and F. Regazzoni, "A survey on hardware trojan detection techniques," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2015, pp. 2021–2024.

[30] B. Lippmann et al., "Physical and functional reverse engineering challenges for advanced semiconductor solutions," in *Proc. Design, Autom. Test Europe Conf. Exhibit. (DATE)*, 2022, pp. 796–801.
[31] D. Burian, "Automated stitching for scanning electron microscopy images of integrated circuits," Diploma thesis, Institut Inf. Syst. Eng., TU Wien, Vienna, Austria, 2022.
[32] R. Courtland, "3D X-ray tech for easy reverse engineering of ICs [news]," *IEEE Spectr.*, vol. 54, no. 5, pp. 11–12, May 2017.
[33] M. Li, B. Yu, Y. Lin, X. Xu, W. Li, and D. Z. Pan, "A practical split manufacturing framework for trojan prevention via simultaneous wire lifting and cell insertion," in *Proc. 23rd Asia South Pacific Design Autom. Conf. (ASP-DAC)*, 2018, pp. 265–270.
[34] Y. Liu, Y. Jin, A. Nosratinia, and Y. Makris, "Silicon demonstration of hardware trojan design and detection in wireless cryptographic ICs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 4, pp. 1506–1519, Apr. 2017.
[35] R. M. Rad, X. Wang, M. Tehranipoor, and J. Plusquellic, "Power supply signal calibration techniques for improving detection resolution to hardware trojans," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2008, pp. 632–639.
[36] P.-S. Ba, S. Dupuis, M. Palanichamy, M.-L. Flottes, G. Di Natale, and B. Rouzeyre, "Hardware trust through layout filling: A hardware trojan prevention technique," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, 2016, pp. 254–259.
[37] K. Xiao and M. Tehranipoor, "BISA: Built-in self-authentication for preventing hardware trojan insertion," in *Proc. IEEE Int. Symp. Hardware-Oriented Security Trust (HOST)*, 2013, pp. 45–50.

Tiago D. Perez received the M.S. degree in electric engineering from the University of Campinas, São Paulo, Brazil, in 2019. He is currently pursuing the Ph.D. degree with the Tallinn University of Technology, Tallinn, Estonia.

From 2014 to 2019, he was a Digital Designer Engineer with Eldorado Research Institute, São Paulo. His fields of work include digital signal processing, telecommunication systems, and IC implementation. His current research interests include the study of hardware security from the point of view of digital circuit design and IC implementation.

Samuel Pagliarini (Member, IEEE) received the Ph.D. degree from Télécom ParisTech, Paris, France, in 2013.

He has held research positions with the University of Bristol, Bristol, U.K., and also with Carnegie Mellon University, Pittsburgh, PA, USA. He is currently a Professor of Hardware Security with Tallinn University of Technology, Tallinn, Estonia, where he leads the Centre for Hardware Security. His current research interests include many facets of digital circuit design, with a focus on circuit reliability, dependability, and hardware trustworthiness.