Hardware Trojan Insertion in Finalized Layouts: a Silicon Demonstration

Tiago Perez, Samuel Pagliarini
Department of Computer Systems - Tallinn University of Technology, Tallinn, Estonia
Emails: {tiago.perez, samuel.pagliarini} @taltech.ee

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 a 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 a 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 65nm CMOS technology comprising 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 SCT against skews in the manufacturing process.

Index Terms—hardware security, manufacturing-time attack, hardware trojan horse, side-channel trojan, VLSI, ASIC.

I. INTRODUCTION

Since the conception of the first integrated circuit (IC), performance increase has been a relentless goal, with a continuous push of the laws of physics to achieve faster and denser ICs. For example, the largest commercial graphic processor unit available in 2021, Nvidia’s GA100 Ampere, contains 59 billion transistors. It is almost 26 million times larger than one of the first commercial microprocessors, Intel’s 4004 with only 2250 transistors, launched in 1971. As the size of the transistors shrinks, the challenge to design and manufacture ICs increases significantly. The ever-increasing cost to build high-end semiconductor manufacturing facilities – building a 3nm 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.

The practice of outsourcing can potentially affect the trustworthiness of an IC as a foundry (or a rogue element within the foundry) can manipulate the design for its own malicious purposes [2]. Many fabrication-time threats have been studied recently, including overproduction, counterfeiting, reverse-engineering, and tampering [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] and IC Camouflaging [7]. 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 [8]. 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 [10], [11]. 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 [12]–[20], 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 [21]. 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. [12] 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.

SCTs are designed to intentionally induce a side-channel to leak confidential information, thus facilitating a SCA. In [13], two lightweight SCT architectures are proposed, both with the intent to induce power consumption in order to leak a crypto key. The first architecture makes use of an adapted code-division multiple access (CDMA) scheme to distribute the leakage of single bits over time (expressed in clock cycles). The modulated bits are forwarded to a special unit (called “leakage circuit”) that creates a CDMA channel over the power side-channel. The second architecture, in addition
to the CDMA scheme, also implements intermediate states within the AES key schedule to facilitate a differential power analysis (DPA) attack. Both architectures are implemented in a field programmable gate-array (FPGA) to demonstrate the capability of the SCT to leak a crypto key. A similar approach that exploits a wireless communication protocol is proposed in [19], in which the authors propose an SCT for orthogonal frequency division multiplexing (OFDM) based wireless crypto cores. Their SCT takes advantage of some properties of the OFDM payload for leaking the crypto key through the transmitted power signal. This approach was tested (in simulation only) utilizing an AES-128 core as the target.

A silicon validated trojan and its architecture is presented by the authors of [14], [15], [20]. The first, presented in [15], is a demonstration of a wireless cryptographic IC (composed of an AES core and an Ultra-WideBand transmitter) that leaks a crypto key after every 128-bit block (ciphertext) is sent by the transmitter. In [20], the authors utilized two prototypes using different technologies for assessing their methodology – 90nm and 65nm. They proposed a parametric SCT aiming to be inserted in SCA-resilient designs in order to breach their defenses. The demonstration is done utilizing a SCA-resilient Present crypto core as a target and, after the SCT is triggered, a power side-channel is enabled for leaking the cryptokey. To broaden the scope of SCTs from dedicated crypto hardware to general-purpose processors (GPPs), an interesting architecture is described in [13], where software models of traditional crypto standards (AES and RSA) are executed on GPPs. Moreover, a number of simple micro-architectural modifications have been described to induce information leakage via faulty computations or variations in the latency and power consumption of certain instructions.

Despite the encouraging results reported from the SCT studies mentioned so far, no study discusses how SCTs could be inserted from the perspective of the attacker. This is also generally true for other types of HTs. In this work, we present not only an SCT design methodology, but also a novel framework for SCT insertion. 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 contributions of this work are a full framework for inserting SCTs, an ASIC prototype for validating our methodology, and also a rich discussion regarding its effectiveness. We fabricated a testchip comprising of 4 cryptocores in a 65nm commercial technology, making our technique silicon proven. Our in-depth analysis of our SCA and the variation in the manufacturing process demonstrated that our SCT is robust against this type of skew. On top of that, we include an analysis of the attack time required for inserting our SCT in a finalized layout when utilizing our ECO flow methodology. These characteristics sharply contrast our work with prior art.

II. THREAT MODEL AND ATTACKER CAPABILITIES

In this work, the principal attacker we are concerned with is a rogue element inside the foundry. He/she has the objective of inserting malicious logic in a finalized layout. 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 is capable of identifying the presence of a crypto core in a layout, which is a reasonable assumption specially for well-known AES implementations that display regularity (due to the round-based key schedule structure). To be very clear, we do not assume the adversary understands the entire victim’s design (nor there is a need for such knowledge). Instead, we assume the adversary can recognize the layout/structure of a single crypto core within a larger design, in line with the assumptions made in [13], [15].

Furthermore, we also assume the adversary: 1) is versed in IC design, 2) enjoys access to modern EDA tools, 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 [15], [16] – 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\(^1\))

\(^1\)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.
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 impractical. Our threat model, therefore, assumes that the attacker only has access to the finalized layout (which is the norm when outsourcing IC fabrication) – he/she would not be able to insert the malicious logic by replicating the physical implementation flow.

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: a technology library, a cell library, the gate-level netlist, and 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 [22], 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 proposed SCT is designed for creating 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 \( N_{\text{leak}} \) leaking bits. Each branch of our RO has two active path options: a direct connection to the next branch or a series of delay cells. Thus, each set of the \( N_{\text{leak}} \) 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.

| 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   |

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 \( N_{\text{key}} \) 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: clock divider (DV), the trojan controller (TC), and the RO. 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 trojan controller cannot match, so to avoid timing violations in the HT itself, clock dividing proves useful. Thus, the clock_set 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, and the power, on its turn, 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:

Fig. 2: Our trojan insertion methodology for a SCT capable of leaking 2 bits per power signature reading (modified from [9]).
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 [22].

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.

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 vector.

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 these, 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. Thus, to model the amplitude steps required for the RO, we need to carefully model its dynamic power consumption.

\[
P_{\text{total}} = P_{\text{static}} + P_{\text{dynamic}}
\]

\[
P_{\text{dynamic}} = \frac{1}{2} V_{DD}^2 F_{sa} \sum_{in} C_{\text{load}}(i) + F_{sa} \sum_{cell_j} E(j)
\]

\[
F_{sa} = 2 F_{RO} = \frac{1}{\tau_{\text{chain}}}
\]

Dynamic power can be calculated using (2), where \(C_{\text{load}}\) 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).

The total path delay of the RO is estimated using (4), where \(\tau_{\text{delay}}\) is proportional to the number of delay cells in the active path times the delay of each cell (\(\tau_{\text{cell}}\), see (6)). The \(\tau_{\text{inverter}}\) delay is from the inverter cells in the feedback path, described by (6). The \(\tau_{\text{control}}\) delay is from the logic cells that controls the active paths, described by (7). Since \(\tau_{\text{inverter}}\) and \(\tau_{\text{control}}\) are fixed for a given implementation, \(\tau_{\text{delay}}\) 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.

\[
\tau_{\text{chain}} = \tau_{\text{delay}} + \tau_{\text{inverter}} + \tau_{\text{control}}
\]

\[
\tau_{\text{delay}} = (N_{D1} + N_{D2} + N_{D3} + N_{D4}) \tau_{\text{cell}}
\]

\[
\tau_{\text{inverter}} = N_{\text{inverter}} \tau_{\text{cell}} + \tau_{\text{nand}}
\]

\[
\tau_{\text{control}} = 7 \tau_{\text{and}} + 3 \tau_{\text{or}}
\]

Finally, the equations above give a first-order estimation of the power profile of the RO. However, since the RO will still undergo place and route, cell position and length of wires have to be properly accounted for. This is achieved by utilizing a SPICE-level simulator and is addressed in Section IV.

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. Typically, this feature is used to perform slight modifications in a finalized layout after its manufacturing, referred to as post-mask ECO. A special type of logic cell, called spare cell, is utilized to enable the ECO methodology. Spare cells are typically inserted in commercial ICs and, when needed, are instantiated by the ECO flow. By doing so, a new design can be generated with minimal changes in the fabrication mask set.

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 [23].

According to Fig. 1, the ECO flow is the last step for the SCT insertion. Before the ECO, the attacker has to identify the filler/spare cells and remove them to create the gaps needed for his own logic. This is achieved by literally one command in physical synthesis tools, as deleting fillers is a typical operation to be performed. 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, if the target circuit performance is perturbed, even if unlikely, it means that the size constraint used for designing the SCT was inappropriate - the adversary then proceeds to pick a different value and leak less bits per clock cycle.
TABLE II: Physical synthesis results for our considered targets, before and after trojan insertion.

| Core          | Frequency (MHz) | Density (%) | Leakage (µW) | Clock Tree Power (µW) | Total Power (µW) | Before SCT insertion | Density (%) | Leakage (µW) | Clock Tree Power (µW) | Total Power (µW) | After SCT insertion |
|---------------|-----------------|-------------|--------------|-----------------------|------------------|----------------------|-------------|--------------|-----------------------|------------------|---------------------|
| AES_LFHD      | 100             | 61          | 77.4         | 115.2                 | 1670             | 63.45                | 80          | 115.8       | 1720                  |                  | \[1720\]             |
| AES_LFHHD     | 100             | 75          | 75.8         | 116.7                 | 1660             | 78.20                | 79          | 117.6       | 1720                  |                  | \[1720\]             |
| AES_HFHD      | 1000            | 58          | 1048         | 1228                  | 22800            | 59.37                | 1052        | 1238        | 23015                  |                  | \[23015\]            |
| AES_LFHFD     | 1000            | 72          | 1036         | 1241                  | 22610            | 73.02                | 1040        | 1252        | 22830                  |                  | \[22830\]            |
| PST_LFHD      | 95              | 53          | 14.13        | 32.05                 | 371.3            | 59.37                | 1052        | 1238        | 23015                  |                  | \[23015\]            |
| PST_HFHD      | 95              | 52          | 34.02        | 325.30                | 3744             | 80.26                | 36.96       | 341.5       | 4015                   |                  | \[4015\]             |
| PST_HFHD      | 95              | 69          | 34.13        | 329.10                | 3785             | 80.26                | 36.96       | 341.5       | 4015                   |                  | \[4015\]             |

The attacker also has to check whether the SCT itself has timing violations. If so, the optional clock divider 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 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) \[24\] 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 \[25\].

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 III. In the column titled ‘Acronym’, we define the terminology used for referring to the many variants of the cores. Results from physical synthesis of the considered targets are presented in Table II. A 65nm commercial CMOS technology was utilized to exercise very challenging placement densities (e.g., 75% for AES_LFHDD) and frequencies (e.g., 0.95GHz for PST_HFHD). The values reported are for typical process corner (TT) and a nominal temperature of 25°C.

TABLE III: Naming convention for the crypto cores regarding their frequency and placement density

| Core | Frequency | Placement density | Acronym |
|------|-----------|-------------------|---------|
| AES  | Low       | Low               | AES_LFDD|
| AES  | High      | High              | AES_LFHDD|
| PST  | Low       | High              | PST_LFHD|
| PST  | High      | Low               | PST_HFHD|

B. SCT Design Results

Initially, all studied cores were physically synthesized for the placement and frequency conditions set above. These results are obtained from Cadence Innovus and are reported as pre-ECO results in Table II (“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 power in our results as it has to be accounted for in the SCT power constraint. Notice how the clock tree power is significant when compared to the leakage power of the targets, 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 clock tree power. 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 a Monte Carlo (MC) simulation using Cadence Spectre. The MC simulation was performed for 10000 samples, varying only the process characteristics, with the temperature fixed to 25°C. Fig. 3 depicts the static power distribution of the PST_HFHD core. As expected, there are obvious outliers. Nevertheless, the distribution average matches the value reported in Table II 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 II 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 the associated power values that effectively leak the key). For this goal, we have taken each of our SCTs and performed custom simulations using Cadence Spectre. The...
### TABLE IV: RO operating frequency and power consumption for four variants of AES and PST.

| Target | RO     | Power & Frequency (µW & MHz) |
|--------|--------|-----------------------------|
|        | core   | S0=0 | S0=1 | S1=0 | S1=1 |
| AES_LF | RO_D6110 | 198@551 | 182@483 | 161@390 | 140@300 |
| AES_HF | RO_D1010 | 196@551 | 182@483 | 161@390 | 140@300 |
| PST_LF | RO_D14  | 16@112 | 11@58 | 10@39 | 8@20 |
| PST_HF | RO_D110 | 32@79  | 36@61 | 31@46 | 26@31 |

![Fig. 4: Side channel trojan functionality example for the AES_LFHD, where the SCT utilizes the RO_D6110 (modified from §).

![Fig. 5: Comparison of area and number of cells between SCTs (from §).

**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. The results for insertion are described on the right side of Table II (‘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 less routing resources, we verified that the ECO flow purposefully utilizes the least congested metal layers. This trend is noticeable in Table V, 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 [26], so overheads in M5 and above are unlikely to affect critical paths.

**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     |

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 of the SCT increased their total density by a small margin. On

---

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 [9]).

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 II. 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 (20ps) 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, regardless of its frequency.

V. TESTCHIP 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 a hardware trojan, 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 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 5 clocks domains: the “always-on” clock is delivered by the signal CLK_CU, and the other 4 domains are connected to the signal CLK_CORE (which assumes 4 different frequencies). For switching the cores on/off selectively, the signals PS1, PS2, PS3 and, PS4 are utilized as described in Tab. VI. Both signals DBG_IN and DBG_OUT are used for debugging purposes only. The power-ground scheme has 2 power sources (VDD and VDDIO) and a common ground (VSS): VDD supplies the core cells with a nominal voltage of 1.0V and VDDIO supplies the IO cells with a nominal voltage of 3.0V.

For manufacturing the chip, we have utilized a commercial 65nm 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 Tab. VI.

The ideal ROs designed in the previous section (see Tab. 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 Tab. 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. 12. 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 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. 9 and are contrasted

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

Fig. 8: Top level diagram (top panel) and floorplan of our testchip (bottom panel).

TABLE VII: RO operating frequency and power consumption for each crypto core of the ASIC prototype.
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. 10 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 Tab. 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 Tab. VI and its distribution across the samples is depicted in Fig. 11. These results are in line with the expected from the power reports done during the physical implementation, as these results are contrasted in Fig. 11 for the worst, typical, and, best-case scenarios (SS-0.9V-0°C, TT-1V-25°C, FF-1.1V-125°C, respectively). The corners provided by the vendor are for extreme cases, i.e., the best-case scenario is characterized at 125°C with an over voltage of 1.1V, in our case the measurements were done at room temperature and at a nominal voltage of 1.0V. Our samples are skewed towards 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:

- A cryptokey with the 8 first bits set to “11-10-01-00” was programmed in the Control Unit’s register bank
- A command for a single encryption was issued
- A command for a single decryption was issued

Right after the encryption is done, the chip asserts one of the “DONE” outputs to mark the time at which the RO starts operating
- 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
- During the whole procedure, the current consumed by the chip is monitored

An example of this procedure for the AES_LFHD core is shown in Fig. 10, 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. 10 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 3 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 Tab. VI) 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 slowest ROs 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

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

Fig. 10: Setup used for bringing up the testchip. 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. 11: Leakage distribution for each crypto core contrasted with the leakage from physical synthesis report for 3 corner cases, and, the leakage of outlier samples.
to make extra considerations for implementing a 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 4 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 2µA precision. Furthermore, the induced power consumption for the smaller crypto cores (PST_LFHD) was in the 20µW range. This makes our SCT a perfect fit for targeting designs that consume very low power while maintaining a reasonable level of stealthiness.

Fig. 12: Our bare die (right) and its layout (left). The lower-right corner is identified by the highlighted pin.

VI. DISCUSSION

For a 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 [27]. 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. Methods for inspecting ICs are separated into two classes, invasive and non-invasive. Invasive methods are generally done by delaminating the IC to reconstruct the layout layers, which leads to the destruction of the inspected sample. Our SCT is likely to be detected by an invasive method due to its size and amount of connected wires. However, these techniques are incredibly time consuming and also require costly and precise equipment. We emphasize that it is not a standard practice of the IC industry to perform this type of analysis.

Differently, non-invasive methods include analyzing physical characteristics of the IC, and/or, the behavior of the IO signals (i.e., timing and state) [10]. 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 [28], 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 [29], [30]. 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. In a future work, we are planning to automate the search for these nodes by submitting the netlist to a high-level functionality reconstruction tool [31]. Such tools can be used to reconstruct the finite-state machine of the target design and/or to give a score to each node in the circuit, thus distinguishing between control and data paths nodes. If a database of known cryptocores can be established in this manner, their localization would become a much simpler task.

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. 11), the SCA was successful (see Fig. 13). When compared with a similar approach presented in [15], 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 clock divider. 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 [32] with the intent to reduce the empty spaces where – potentially – a 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 trade-off between security and power consumption not interesting for many vendors and applications. Therefore, the adoption of this technique 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 [14], [15], [20], 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 re-implementing the design has a significant runtime, in the case of our testchip (see Fig. 9) it is required at least 7 hours and 18 minutes. 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 testchip is only 1 hours and 11 minutes. 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 Secion 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. CONCLUSIONS

In order to steal secret information from crypto-capable ICs, a rogue element within the foundry may insert a side-channel trojan. 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 our SCT. 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 testchip, all the 4 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 hardware trojans in order to understand if a capable adversary is able to execute a “blind” yet successful insertion, i.e., one that does not require prior information and/or knowledge about the targeted core.

ACKNOWLEDGMENT

This work has been partially conducted in the project “ICT programme” which was supported by the European Union through the European Social Fund. It was also partially supported by the Estonian Research Council grant MOBERC35.

REFERENCES

[1] M. Lapedus, “Big trouble at 3nm,” [Online]. Available at: https://semiengineering.com/big-trouble-at-3nm/.
[2] U. Guin et al., “Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain,” Proceedings of the IEEE, vol. 102, no. 8, pp. 1207–1228, 2014.
[3] S. M. Ben, “Security challenges and requirements for industrial control systems in the semiconductor manufacturing sector,” 2012.
[4] D. 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 GLSVLSI ’19, p. 471–476, 2019.
[6] M. Yasin, J. Rajendran, and O. Sinanoglu, Trustworthy Hardware Design: Combinational Logic Locking Techniques. Springer, Cham, 2019.
[7] M. Li, K. Shamsi, T. Meade, Z. Zhao, B. Yu, Y. Jin, and D. Z. Pan, “Provable secure camouflaging strategy for ic protection,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 38, no. 8, pp. 1399–1412, 2019.
[8] P. Subramanayam, S. Ray, and S. Malik, “Evaluating the security of logic encryption algorithms,” in 2015 IEEE HOST, pp. 137–143, 2015.
[9] T. Perez, M. Imran, P. Vaz, and S. Pagliarini, “Side-channel trojan insertion - a practical foundry-side attack via eco,” in 2021 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–5, 2021.
[10] M. Tehranipoor and F. Koushanfar, “A survey of hardware trojan taxonomy and detection.” IEEE Design and Test of Computers, vol. 27, no. 1, pp. 10–25, 2010.
[11] M. Rostami, K. Koushanfar, and R. Karri, “A primer on hardware security: Models, methods, and metrics.” Proceedings of the IEEE, vol. 102, no. 8, pp. 1283–1295, 2014.
[12] L. Lin, W. Burleson, and C. Paar, “Moles: Malicious off-chip leakage enabled by side-channels,” in 2009 IEEE/ACM International Conference on Computer-Aided Design, pp. 117–122, 2009.
[13] L. Lin et al., “Troyan side-channels: Lightweight hardware trojans through side-channel engineering,” in Cryptographic Hardware and Embedded Systems - CHES 2009, pp. 382–395, 2009.
[14] Y. Jin and Y. Makris, “Hardware trojans in wireless cryptographic ics,” IEEE Design Test of Computers, vol. 27, no. 1, pp. 26–35, 2010.
[15] Y. Liu, Y. Jin, and Y. Makris, “Hardware trojans in wireless cryptographic ics: Silicon demonstration & detection method evaluation,” in Int. Conf. on Computer-Aided Design (ICCAD), pp. 399–404, 2013.
[16] R. Kumar, P. Jovanovic, W. Burleson, and I. Polian, “Parametric trojans for fault-injection attacks on cryptographic hardware,” in 2014 Workshop on Fault Diagnosis and Tolerance in Cryptography, pp. 18–28, 2014.
[17] K. Yang, M. Hicks, Q. Dong, T. Austin, and D. Sylvester, “A2: Analog malicious hardware,” in 2016 IEEE Symposium on Security and Privacy (SP), pp. 18–37, 2016.
[18] J.-F. Gallais et al., “Hardware trojans for inducing or amplifying side-channel leakage of cryptographic software,” in Trusted Systems, pp. 253–270, 2011.
[19] L. Ali and Farshad, “Analog hardware trojan design and detection in OFDM based wireless cryptographic Ics,” Plos One, vol. 16, no. 7, p. e0254903, 2021.

S. Ghandali, T. Moos, A. Moradi, and C. Paar, “Side-Channel Hardware Trojan for Provably-Secure SCA-Protected Implementations,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 28, no. 6, pp. 1435–1448, 2020.

P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” in Advances in Cryptology — CRYPTO ’98 (M. Wiener, ed.), pp. 388–397, 1999.

R. S. Rajaratnam, Y. Lin, Y. Jin, and D. Z. Pan, “Regus: A reverse engineering framework from gdii to gate-level netlist,” in 2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pp. 154–163, 2020.

T. Trippel et al., “ICAS: An Extensible Framework for Estimating the Susceptibility of IC Layouts to Additive Trojans,” 2020 IEEE Symposium on Security and Privacy (SP), pp. 1078–1095, 2020.

A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, “Present: An ultra-lightweight block cipher,” in Cryptographic Hardware and Embedded Systems - CHES 2007 (P. Paillier and I. Verbauwhede, eds.), (Berlin, Heidelberg), pp. 450–466, Springer Berlin Heidelberg, 2007.

S. Ghandali et al., “Side-channel hardware trojan for provably-secure sca-protected implementations,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 28, no. 6, pp. 1435–1448, 2020.

S. N. Pagliarini, M. M. Isgenc, M. G. A. Martin, and L. Pileggi, “Application and product-volume-specific customization of boeul metal pitch,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 9, pp. 1627–1636, 2018.

S. Bhasin and F. Regazzoni, “A survey on hardware trojan detection techniques,” in 2015 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 2021–2024, 2015.

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 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 265–270, 2018.

Y. Liu, Y. Jin, A. Nosratinia, and Y. Makris, “Silicon demonstration of hardware trojan design and detection in wireless cryptographic ics,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 4, pp. 1506–1519, 2017.

R. M. Rad, X. Wang, M. Tehranipoor, and J. Plusquellec, “Power supply signal calibration techniques for improving detection resolution to hardware trojans,” in 2008 IEEE/ACM International Conference on Computer-Aided Design, pp. 632–639, 2008.

T. Meade, S. Zhang, and Y. Jin, “Netlist reverse engineering for high-level functionality reconstruction,” in 2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 655–660, 2016.

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 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp. 254–259, 2016.

Tiago D. Perez received the M.S. degree in electric engineering from the University of Campinas, S ˜ao Paulo, Brazil, in 2019. He is currently pursuing a Ph.D. degree at Tallinn University of Technology (TalTech), Tallinn, Estonia. From 2014 to 2019, he was a Digital Designer Engineer with Eldorado Research Institute, S ˜ao Paulo, Brazil. 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 (M’14) received the PhD degree from Telecom ParisTech, Paris, France, in 2013. He has held research positions with the University of Bristol, Bristol, UK, and with Carnegie Mellon University, Pittsburgh, PA, USA. He is currently a Professor of Hardware Security with Tallinn University of Technology (TalTech) in 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.