ABSTRACT
Since the inception of the integrated circuit (IC), the size of the transistors used to construct them continually shrink. While this advancement significantly improves computing capability, the associated massive complexity forces IC designers to outsource fabrication. Outsourcing presents a security threat: comprehensive post-fabrication inspection is infeasible given the size of modern ICs, thus it is nearly impossible to know if the foundry has altered your design during fabrication (i.e., inserted a hardware Trojan). Defending against a foundry-side adversary is challenging because—with as little as two gates—hardware Trojans can completely undermine software security. Prior work attempts to both detect [2, 24, 40, 44, 51, 54] and prevent [4, 5, 60] such foundry-side attacks, but all existing defenses are ineffective against the most advanced hardware Trojans [61].

We present Defensive Routing (DR), a preventive layout-level defense against untrusted foundries, capable of thwarting the insertion of even the stealthiest hardware Trojans. DR is directed and routing-centric: it prevents foundry-side attackers from connecting rogue wires to security-critical wires by shielding them with guard wires. Unlike shield wires commonly deployed for cross-talk reduction, DR guard wires present an additional technical challenge: they must be tamper-evident in both the digital and analog domains. To address this challenge, we present two different categories of guard wires: natural and synthetic. Natural guard wires are comprised of pre-existing wires that we route adjacent to security-critical wires, while synthetic guard wires are added to the design specifically to protect security-critical wires. Natural guard wires require no additional hardware and are digitally tamper-evident. Synthetic guard wires require additional hardware, but are tamper-evident in both the digital and analog domains.

We implement automated tools for deploying both types of guard wires in IC layouts of commercial complexity. We evaluate the protections provided by both natural and synthetic guard wires across three different IC designs: a processor and AES and DSP accelerators. We then compare the efficacy of DR to existing placement-centric layout-level defenses. DR is shown to successfully defend against even the stealthiest hardware Trojans, across several designs, with less than 1% power, performance, and area overheads.

KEYWORDS
Hardware Security; Fabrication-time Attacks and Defenses; VLSI

1 INTRODUCTION
Integrated circuits (ICs) are the foundation of computing systems. Security vulnerabilities in silicon are devastating as they subvert even formally verified software. For almost 50 years, the transistors within ICs have continued to shrink, enhancing performance while reducing power and area usage. However, these advances that push the laws of physics come with a financial cost: the price to build a 3nm fabrication facility capable of producing ICs at a commercial scale is estimated to be $15–20B [30]. Even when entities can afford to make such an investment, they must continually run the IC fabrication line (approximately 40,000 wafers/month) as many fabrication process cannot be readily stopped and restarted.

This extreme cost forces most semi-conductor companies, and even nation states, to become “fabless”, i.e., they outsource fabrication. As of late August 2018, only 3 companies in the world (Intel, Samsung, and TSMC) have capabilities to fabricate ICs at the 10/7nm process nodes [31]. This presents a security threat: fabless semiconductor companies and nation states must trust these three manufacturers (and their partners) not to alter their designs at any point throughout the fabrication process (i.e., implant an attack).

The most stealthy and controllable fabrication-time attacks involve inserting additional \(^1\) circuit components designed to maliciously subvert the functionality of the chip (i.e., an additive hardware Trojan). Until recently, fabrication-time attacks were only contrived and demonstrated in literature. In October 2018, Bloomberg

---

\(^1\)Additive hardware Trojans are a class of Trojan designs which require additional hardware to be added to a circuit design. The only non-additive Trojans that have been conceived are the dopant-level Trojans [9, 28], which have limited controllability and are detectable [51].
Businessweek published an article detailing a fabrication-time attack discovered in the wild [45]. This attack was carried out at the printed circuit board (PCB) level and involved adding a rogue IC to server motherboards designed by Supermicro. The rogue IC was placed on the motherboard such that it could tamper with memory writes. While fabrication-time attacks at the PCB level are non-trivial to detect, fabrication-time attacks carried out at the IC level are even stealthier. Circuit components at the IC level are orders-of-magnitude more numerous and smaller than PCB level components, i.e., billions vs. tens and mms vs. mms.

To date, there are two ways of defending against fabrication-time attacks: post-fabrication detection and pre-fabrication prevention. The former tries to detect the presence of hardware Trojan components through various means after the chip has been fabricated, while the latter attempts to alter an IC’s physical layout, at design time, in a way that makes it challenging for a foundry-side attacker to modify.

Detection is more commonly studied than prevention and consists primarily of two techniques [54]: 1) side-channel analysis and 2) functional testing. Side-channel analysis attempts to detect noticeable deviations in power usage, electromagnetic (EM) emanations, performance (timing), etc. [2, 24, 40, 44]. It often requires a “golden” reference chip to be effective, and can only detect the side-channel signature deviations greater than those caused by process variation (i.e., the hardware Trojan must have a large physical footprint). Alternatively, functional testing attempts to inadvertently trigger the Trojan by activating as many logic paths through the circuit as possible. Functional testing does not require any “golden” reference chip, but it requires the Trojan’s trigger to be activated by the IC’s common mode operation, as exhaustive testing of even a moderately complex integrate circuit is infeasible.

As the limitations of existing detection-based approaches become clear [61], prevention-based defenses against hardware Trojans have recently emerged. To the best of our knowledge, only three preventive hardware Trojan defense mechanisms have been explored [4, 5, 60]. All three are placement-centric, attempting to increase the device layer (core) density by filling open spaces with interconnected functional logic gates. Thus, making it challenging for an attacker to find open space in the design to insert his/her Trojan logic gates. However, there are several problems with placement-centric defenses. As Ba et al. [5] point out, the BISA cell approach proposed by Xiao et al. [60] is infeasible as it requires 100% placement density. Contrast this with the sub-70% density of current IC layouts that ensures routability. If 100% density were feasible, every IC design would be manufactured that way to save cost. Alternatively, Ba et al. [4, 5] suggest targeted filling: only filling placement sites that are located closest to “security-critical” logic with tamper-evident logic. However, as shown in our evaluation, this defense can prevent the insertion of large Trojans, or Trojans with tight timing constraints.

Unfortunately, no single technique is effective at detecting, and/or preventing, the insertion of the stealthiest known additive hardware Trojan [61]. To fill this gap in current defenses, we propose Defensive Routing (DR), a routing-centric defense that aims to prevent foundry-side attackers from attaching attack logic to security-critical wires. We define DR as any routing method that protects security-critical wires from fabrication-time alterations. Specifically, we leverage concepts from the signal-integrity domain [20, 21] and apply them to a security domain (addressing several technical challenges along the way): we route “guard wires” around security-critical wires that make it infeasible for an attacker to tap any security-critical wire without detection (i.e., tamper evident), something characteristic of additive Trojans (Fig. 1). Extending signal-integrity domain techniques to the security domain entails two technical challenges:

1. completely cover the surface area of security-critical wires,
2. and be tamper-evident.

Additionally, contrary to placement-centric defenses, which focus on preventing attack implementation, DR focuses on preventing attack integration, thus it does not require filling all the empty space in an IC design to be effective.

We make the following main contributions:

- The first routing-centric, preventative, defense to IC fabrication-time attacks. It routes tamper-evident guard wires alongside security-critical wires, making fabrication-time modifications to such wires infeasible and/or detectable post-fabrication.
- Introduction and use of natural and synthetic guard wires. Natural guard wires are inherent to the functionality of the design that are routed nearby security-critical wires. Synthetic guard wires are not inherent to the design, but are added during the place-and-route phase of the IC design process. Using synthetic guard wires, we demonstrate the possibility of completely guarding all security-critical wires in a design.
- Design and evaluation of tamper-detection mechanisms that expose guard wire modifications.
- Automated tool-chain for inserting synthetic and natural guard wires that integrates with existing commercial and open-source VLSI CAD tools.
- Evaluation of the levels of defense provided by both synthetic and natural guard wires across three security-critical IC layouts: 1) a processor, 2) AES and 3) DSP accelerators. We show that DR is more effective than existing placement-centric defenses [4, 5, 60], and is capable of thwarting even the stealthiest hardware Trojans, including A2 [61].

2 BACKGROUND

2.1 IC Design Process

Creating an Integrated Circuit (IC) consisting of a billion transistors is a complex process that requires its decomposition into sub-processes and extensive use of automation via Computer Aided Design (CAD) tools. The IC design process consists of five main phases, as illustrated in Fig. 2. First, during HDL design, high-level descriptions of the IC are written in Hardware Description Languages (HDL) like Verilog or VHDL. Next, during synthesis, the HDL code is “compiled” into a gate-level netlist. The gate-level netlist is then placed-and-routed (PaR) and a physical geometric blueprint of the chip is encoded in a Graphics Database System II (GDSII) file. Lastly, the IC is fabricated, and then packaged into a device for mounting on a printed circuit board. In line with prior foundry-level attack studies [4, 5, 9, 28, 60, 61], and economic forces, we assume all design phases—except fabrication—are trusted.
Defensive routing is deployed at the physical (layout) level, or during the place-and-route (PaR) design phase. During PaR, the gate-level netlist is physically arranged onto a 3-dimensional grid, shown in Fig. 3. The 3D grid consists of a device layer, where circuit components (e.g., digital logic gates) are placed, and several routing layers vertically stacked above, where wires are routed to connect the circuit components on the device layers. Each layer is separated by an insulating dielectric, and vias are used to connect wires on adjacent layers.

2.2 Hardware Trojans

A hardware Trojan is a malicious modification to a circuit designed to alter its operative functionality [8]. It consists of two main building blocks: a trigger and payload [13, 24, 59]. Prior work provides hardware Trojan taxonomies based on the type of trigger and payload designs they employ [13, 24, 59]. We adopt this taxonomy; depicted in Fig. 4.

2.2.1 Trigger. The trigger is circuitry that initiates the delivery of the payload when it encounters a specific state. The goal of the trigger is to control payload deployment such that it is hidden from test cases (stealthy), but readily deployable by the attacker (controllable). Triggers are created by adding, removing, and/or manipulating existing circuit components [28, 47, 54, 61], and can be digital or analog [26, 46, 61]. The ideal trigger achieves stealth and controllability while being small (i.e., requiring few additional circuit components).

2.2.2 Payload. The payload is circuitry that, upon being signaled by the trigger, alters the functionality of the victim (host) circuit.

Like the trigger, the payload can be analog or digital, and has a variety of possible malicious effects. Prior work demonstrates Trojan payloads that leak information [34], alter the state of the IC [61], and render the IC inoperable [47]. One attribute all documented stealthy and controllable hardware Trojans have in common is that they must route a rogue wire to, or directly adjacent to, a security-critical wire within the victim IC.

2.2.3 Fabrication-Time Attacks. Inserting a hardware Trojan at fabrication time is different from inserting a Trojan during the front-end design. Unlike behavioral or structural-level attackers that maliciously modify the HDL or gate-level netlist, respectively [3, 23, 58], the fabrication-time attacker only has access to the physical level representation of the IC design (i.e., output of phase 3 in Fig. 2). Specifically, they must edit the geometric representation of the circuit layout. While this is more challenging than editing the design at the behavioral- (HDL) or structural-level (netlist), where design specific semantics are more readily interpretable, it is even more difficult to defend. The post-fabrication defender receives a literal black box from the foundry. Comprehensively inspecting each fabricated die to verify the absence of malicious perturbations is infeasible for the most advanced hardware Trojans [61].

Inserting a hardware Trojan into a target IC design at fabrication-time involves two steps: Trojan placement and Trojan routing [61]. The former is the process of finding empty space on the IC’s device layer to add additional circuit components, e.g., logic gates, to construct the Trojan trigger and payload components. Trojan routing involves both routing the trigger and payload circuit components together and routing the trigger/payload circuitry to a target security-critical wire(s). Prior work increases the difficulty of Trojan placement by filling empty spaces on the IC’s device layer with functional logic gates [4, 5, 60]. While effective for Trojans with large footprints, filling all placement sites is infeasible [5], leaving the IC still vulnerable to Trojans with small footprints [61]. Orthogonally, DR aims to increase the difficulty of Trojan routing by protecting security-critical wires.

2.3 Interconnect Models

The are two main ways to model IC wires (interconnect): lumped and transmission-line models [6]. Lumped interconnect models approximate interconnects using networks of resistors and capacitors.
Transmission-line models approximate interconnects as transmission lines with a characteristic impedance and propagation delay.

The choice of interconnect model is a function of maximum frequency component to wire length [52]. A common rule of thumb for IC interconnects is: a wire is considered a transmission line if its length is greater than \( \approx 10\% \) of the wavelength of the maximum frequency component it transmits [52]. In digital electronics, it is common to think of signals in terms rise and fall times, rather than maximum frequency component. Thus, one can modify the prior rule of thumb to: a wire is considered a transmission line if the transmitted signal rise time, \( T_{rise} \), is less than twice the wire’s propagation delay, \( T_{pd} \) [52]. Eq. (1) captures this rule of thumb.

\[
Model = \begin{cases} 
\text{Transmission Line,} & T_{rise} < 2T_{pd} \\
\text{Lumped RC,} & \text{otherwise}
\end{cases}
\] (1)

Choosing the right model is vital to understanding operational limitations and ensuring signal integrity within an IC design. For example, an interconnect that carries high-speed signal transitions will observe signal reflections from impedance discontinuities that are destructive to the signal integrity of the overall system. Modeling such interconnects using a lumped RC model can hide these destructive effects, while a transmission-line model would not.

### 2.4 Time-Domain Reflectometry (TDR)

Time-domain reflectometry (TDR) is a transmission line analysis technique that involves injecting a single rising pulse down a transmission line and analyzing its reflection(s). TDR enables us to deduce many characteristics about the device under test (DUT)—the transmission line—such as impedance discontinuities, propagation delays, dielectric constant, and so on [17, 22]. By Eq. 1, the faster the rising edge of TDR’s incident pulse, the finer-grain of propagation delay changes are detectable. TDR was first developed as a fault-analysis technique for long transmission lines, such as telephone or optical communication lines [43, 49]. As commercial TDR systems became more advanced, TDR became a standard IC packaging fault analysis tool [14, 41, 48]. Researchers have now demonstrated terahertz level TDR systems capable of locating faults in IC interconnects to nanometer scale accuracies [12, 39, 53, 55]. With such fine-grain resolution, TDR is an ideal tamper-analysis technique for ensuring the integrity of the guard wires used in DR.

### 3 THREAT MODEL

We adopt a threat model in which all phases of the IC design process are trusted except fabrication (Fig. 2). The untrusted foundry threat model stems from the extreme ramp-up costs associated with fabricating leading-edge silicon [30, 31] that make outsourcing IC fabrication a necessity—even for nation states. In line with similar threat models [46, 54, 61], we assume that any fabrication-time modification(s) are carried out by a malicious actor within the foundry (or any foundry partners) that has access only to the physical representation of the IC in the form of a GDSII file. The adversary can see and modify the entire IC layout.

While there are many types of hardware Trojans [46] (§ 2.2), we focus on additive Trojans, rather than subtractive or substitution Trojans. Additive Trojans require adding additional circuit components and wiring to the IC design. We focus on additive Trojans since the only non-additive hardware Trojans conceived are the dopant-level Trojans [28, 47], but both have limited controllability and are detectable with optical microscopy [51].

To successfully implement an additive hardware Trojan, the adversary must complete both Trojan placement and Trojan routing, without being exposed. Namely, they must 1) find empty space on the device layer to insert the Trojan’s trigger and payload components, e.g., logic gates, and 2) locate an unblocked segment on a security-critical wire to attach the Trojan components to. They are restricted from modifying the dimensions of the chip and/or violating manufacturing design rules that would risk exposure. They are allowed to move components and/or existing wiring around, but are constrained by available resources (e.g., time) and correctness from making mass perturbations to the layout. As process technologies scale, manufacturing design rules become increasingly complex [50]. Thus, rearranging components and/or existing wiring comes at a substantial cost. The time to complete any layout modifications, and verify such modifications have not violated design correctness, cannot disrupt the fabrication turn-around time expected by their customers.\(^2\) Additionally, the attacker avoids any modifications that are detectable using existing test case- or side-channel-based defenses. While it is trivial for an attacker with infinite time and resources to reverse engineer the physical layout into HDL, add a Trojan, and re-run the design through the entire IC design process (Fig 2)—generating an entirely new layout—we assume such an attack is infeasible in the hard time limits of fabrication contracts, thus outside the scope of our threat model.

### 4 DEFENSIVE ROUTING (DR)

The two steps of inserting a hardware Trojan at fabrication-time are Trojan placement and Trojan routing (§ 2.2.3). Thus, the two logical approaches for defending against fabrication-time attacks are centered around making Trojan placement and/or Trojan routing challenging for an adversary. Specifically, DR takes a directed approach to make Trojan routing intractable by eliminating any potential interface points between a hardware Trojan and the victim IC. We leverage the common observation that only a subset

\(^2\)Typically, fabrication turn-around times are approximately 3 months [29, 57].
While re-route attacks are difficult to detect, they are also more wires, thus creating an additional obstacle for adversaries to overcome. Guard wires shield the surfaces of security-critical wires to connect a rogue Trojan wire to a security-critical wire: A) delete (layer jumping connection), as shown in Fig. 5c. Such an attack Trojan. This observation enables the practicality of DR.

Therefore, our challenge is to design guard wires that are tamper-evident. Deletion and cutting attacks are trivial to detect with a continuity test. Re-route attacks, however, are more challenging to detect. Detecting a re-route attack requires the ability to measure physical characteristics (e.g., length) about a guard wire post-fabrication. While re-route attacks are difficult to detect, they are also more difficult to implement. Any edits an adversary makes to the layout must not violate manufacturing design rules (§ 3). When the design-density is high, re-route attacks are difficult for an attacker to implement as leading-edge process technologies have thousands of manufacturing design constraints [50].

Based on these attack considerations, we design two types of guard wires with varying tamper-evident properties and deployment costs: natural and synthetic guard wires. We summarize each guard wire type in Fig. 6, and their tamper-evident-properties/deployment-costs in Fig. 7.

4.1 Natural Guard Wires
Natural guard wires (Fig. 6a) are wires inherent to the host IC design that are routed nearby security-critical wires. They create hyper-local routing densities nearby security-critical wires, thus limiting, or eliminating the locations on security-critical wires where an attacker can attach a rogue Trojan wire. By re-purposing pre-existing wires as guard wires, natural guard wires have no hardware overhead. However, there are additional routing constraints (e.g., length, layer, location and spacing) that accompany natural guard wires, thus making deployment challenging, i.e., it is difficult to completely protect all surfaces of all security-critical wires (Fig. 10).

Natural guard wires are inherently tamper-evident with respect to deletion and cutting attacks (Fig. 7). An adversary cannot delete or significantly modify natural guard wires without risking alteration of the functionality and/or operating specifications of the host IC, and thus being exposed. Natural guard wires may also be re-route tamper-evident depending on routing-constraints, e.g., if the natural guard wires are critical paths in the IC (i.e., timing sensitive).

4.2 Synthetic Guard Wires
Synthetic guard wires are not inherent to the host IC design. They are added to the design during the place-and-route IC design phase (Fig. 2). They eliminate the accessible surface area of security-critical wires, blocking an attacker from attaching a Trojan wire at fabrication-time. Synthetic guard wires incur hardware overhead, i.e., additional wires. However, they are easier to deploy as they are not subject to additional routing constraints. Therefore, it is possible to protect all surfaces of all security-critical wires in a design (Fig. 10).

There are several synthetic guard wire architectures that may be deployed, listed in order of increasing difficulty of deployment: 1) fully-disjoint, 2) partially-connected, and 3) fully-connected. Fully-disjoint synthetic guard wires are not connected between sides, i.e., the guard wires on each side of a security-critical wire are never connected to one another. Partially-connected guard wires allow for a single guard wire to be utilized on multiple sides. For example, a security-critical wire could be guarded on the north, east, and west sides by a single guard wire that wraps around the security-critical wires.
wire. Lastly, fully-connected guard wires are formed when a single guard wire is routed around all sides of all security-critical wires.

Unlike natural guard wires, synthetic guard wires are not inherently tamper-evident. To enable post-fabrication tamper analysis (§2.4 and 6.4), the analog characteristics of guard wire segments must be observable. This can be implemented either on-chip, e.g., with internal sensors [25] or ring oscillators [63], or off-chip, e.g., with an I/O pin and a one-time programmable fabric [35].

5 IMPLEMENTATION

We develop an automated tool-chain for deploying DR in modern IC designs. Our toolchain integrates with existing IC design flows (Fig. 2) that utilize commercial VLSI CAD tools. Specifically, we implement the DR toolchain around the Cadence Innovus Implementation System [11], a commercial place-and-route (PaR) CAD tool. The tool-chain is invoked by modifying a place-and-route TCL script,4 as shown in Fig. 8.

5.1 Place-&-Route Process

The PaR design phase (Fig. 2) is typically automated by a CAD tool, programmatically driven by TCL script(s). There are several steps to PaR that are performed in the following order: 1) floorplanning, 2) placement, 3) clock tree synthesis, 4) routing, and 5) filling. To ensure that all guard-wires are routed optimally, we modify the order of these PaR steps. Specifically, after floorplanning (1), we use our automated toolchain to place security critical components and route security-critical and guard-wires. Our toolchain then permanently fixes the locations of these components and wires to prevent the PaR CAD tool from modifying their positions and/or shapes throughout the remainder of the PaR process. Lastly, we utilize the PaR CAD tool to place non-security-critical components (2), synthesize the clock tree (3), route non-security-critical (4) and fill the design with filler (cap) cells.

5.2 Automated Toolchain

The DR toolchain automates the insertion of either natural or synthetic guard wires around security-critical wires. The toolchain consists of three main phases (Fig. 8). The first phase (A) identifies security-critical nets. The second phase (B) identifies the unblocked surfaces of all security-critical interconnect within a GDSII-encoded layout. The last phase (C) guards the security-critical interconnects by routing guard wires nearby. We provide additional implementation details on all three stages of the DR toolchain below.

5.2.1 Identifying Security-Critical Nets. The first phase of DR requires identifying those nets that are security-critical (i.e., wires to guard). Like prior work [23, 35], we assume the designer has annotated the HDL to flag nets that are to be treated as "security-critical" by appending a unique prefix to such signal names. From here, we perform a circuit-level dataflow analysis to generate signal fan-in graphs from the hand-annotated HDL. Specifically, our tool takes as input a Verilog netlist and the (root signal) annotations, and returns a graph detailing the signal(s) that comprise the fan-in to those root signals, to a configurable depth.

Given the interconnected nature of signals within an IC design, an adversary may elect to target a wire that influences a root security-critical wire, rather than the wire itself. To address this indirection, our tool widens the set of signals deemed “security-critical” beyond the scope of those (hand-annotated) root security-critical signals. Since the netlist is often modified by the PaR CAD tool to meet various design constraints (e.g., power, performance, and area), we invoke our dataflow analysis after all security-critical wires have first been routed (§ 5). Note: while CAD tools can optimize (i.e., modify) the design netlist through the IC design flow, they leave signal name prefixes intact.

5.2.2 Identifying Unblocked Wire Surfaces. The second phase of DR identifying the unblocked surfaces of security-critical interconnect in a physical IC layout, i.e. potential Trojan wire attachment locations. To do so, we use GDS2Score [37], an open-source framework for computing security metrics over an IC layout. GDS2Score

4Tool Command Language (TCL) scripts are the standard programmatic interface to commercial VLSI CAD tools. IC designers often develop a set of scripts for driving the CAD tools that automate most of the IC design process (Fig. 2).
Defensive Routing: a Preventive Layout-Level Defense Against Untrusted Foundries

5.2.3 Guard Unblocked Wire Surfaces. The last stage of the DR toolchain (Fig. 8) is a custom guard wire routing tool. We implement this tool in Python. It takes as input exact locations of security-critical wires and their unblocked sides (output from the GDS2Score net blockage metric) and generates a TCL script that integrates with the Cadence Innovus Digital Implementation platform [11] to automatically route the guard wires. This TCL script is executed immediately after the security-critical wires have been routed, but before placing the remaining non-security-critical components. Depending on the type of guard wires being deployed, natural, synthetic, or both, different guard wire TCL scripts are generated as described below.

Natural Guard Wires. There are numerous ways natural guard wires can be implemented. Since commercial PaR CAD tools do not offer an interface to enable fine-grain constraints between two unrelated signal wires, we develop an indirect method for implementing natural guard wires. We implement natural guard wires by constraining placement and routing resources nearby security-critical wires. First, we identify all circuit components (i.e., logic gates) connected to all security-critical wires, i.e., security-critical components. Next, we draw a bounding box around these components and extend this boundary vertically by 10% of the overall box height, and horizontally by 10% of the overall box width. Then, we set placement and routing density screens in the portion of the IC layout that lies outside the security-critical bounding box. These constraints limit the placement and routing resources outside the bounding box, thus forcing more components and wiring within the bounding box. With increased routing density nearby security-critical wires, they are less accessible by Trojan payload delivery wires. The density screen configuration settings are optimized to maximize the net blockage metric computed by the GDS2Score metric. Fig. 9a depicts natural guard wires deployed around the security-critical privilege bit signal of the OR1200 processor IC design.

Synthetic Guard Wires. Synthetic guard wires are more straightforward to implement. The automated guard wire deployment toolchain locates all unblocked surfaces (north, south, east, west, top, and bottom) of all security-critical wires and routes guard wires in these regions. After all guard wire segments are routed, they are connected according to the architecture chosen (§ 4.2). Fig. 9b depicts synthetic guard wires deployed around the security-critical privilege bit signal of the OR1200 processor IC design.

6 EVALUATION

We evaluate DR in three areas. First, we explore the effectiveness of DR at closing the fabrication-time attack surface of three security-critical IC designs of commercial complexity, comparing the capabilities of DR with state-of-the-art layout-level defenses [4, 5, 60]. Next, we demonstrate the practicality of DR, analyzing its power, performance, and area overheads. Finally, we perform a threat assessment, demonstrating how guard wires are tamper-evident.

6.1 Experimental Setup

6.1.1 IC Designs. We utilize three IC designs for our evaluation: OR1200 processor SoC, AES accelerator, and DSP accelerator. The OR1200 processor SoC is an open-source design [42] used in previous fabrication-time attack studies [61]. The AES and DSP accelerator designs are open-sourced under the Common Evaluation Platform (CEP) benchmark suite [36]. The OR1200 processor SoC consists of a 5-stage pipelined OR1200 CPU that implements the 32-bit OR1K instruction set and Wishbone bus interface. It runs Linux and Busybox. The AES accelerator supports 128-bit key sizes. The DSP accelerator implements a Fast Fourier Transform algorithm.

All designs target a 45nm Silicon-On-Insulator (SOI) process technology. They are synthesized with Cadence Genus (v16.23) and placed-and-routed using Cadence Innovus (v17.1). All designs target a 100 MHz clock frequency and a core density of 60–70%. All ICs are synthesized and placed-and-routed on a server with 2.5 GHz Intel Xeon E5-2640 CPU and 64 GB of memory running Red Hat Enterprise Linux (v6.9).

6.1.2 Root Security-Critical Signals. Signals (nets) are classified as security-critical if they are apart of the fan-in to a set of root security critical signals that are hand annotated by the HDL designer (§ 5.2.1). For the OR1200 processor design, we select the
supervisor signal (supv) as security critical, because altering this signal enables an attacker to escalate the privilege mode of the entire processor, as shown in previous work [18, 26, 61]. For the AES accelerator, we mark the 32-bit key-loading register as security-critical. Attacking the key-loading register of the AES accelerator enables the attacker to leak encryption keys. Lastly, for the DSP accelerator, we mark the the next_out signal as security-critical. The DSP accelerator’s next_out signal indicates to external hardware when an FFT computation is ready at the output registers. Tampering with the next_out signal allows the attacker to hide specific outputs of the chip. Fig. 12 (far right) shows the number of interconnect wires our automated toolchain classifies as security-critical, with the aforementioned root signals marked as security-critical, for each IC layout.

6.2 Effectiveness

We first evaluate the effectiveness of DR in thwarting the insertion of hardware Trojans at fabrication-time. We compare the degree of protection provided by DR with that provided by deploying another layout-level defense suggested by Ba et al [4, 5]. This placement-based defense involves filling as many empty component placement sites as possible (they show that filling 100% of placement sites is infeasible), prioritizing empty sites nearest security-critical nets. We use our automated toolchain (§ 5.2) to deploy both types of guard wires (natural and synthetic). We simulate Ba et al.’s placement defense [4, 5] using a custom GDS2Score extension.

6.2.1 GDS2Score. We use the open-source GDS2Score framework [37] to quantify the effectiveness of each defense. GDS2Score analyzes the physical layout of an IC (encoded in a GSDII file) computes security metrics detailing the IC layout’s fabrication-time attack surface.

GDS2Score provides three metrics: 1) net blockage, 2) trigger space, and 3) route distance. The net blockage metric computes the percentage of surface area of security-critical nets that are blocked by other circuit components or wiring. The trigger space metric computes the amount of unused placement sites on the device layer that may be utilized by an adversary to place a Trojan trigger or payload circuit(s). Lastly, the route distance metric computes the minimal distance between unblocked security-critical nets and unused placement sites that an adversary would have to route a rogue Trojan wire to “connect” the hardware Trojan to the host IC. The trigger space metric details the difficulty of performing Trojan placement, while the net blockage and route distance metrics detail the difficulty of performing Trojan routing. Specifically, we use GDS2Score’s net blockage and route distance security metrics to compare both types of DR guard wires and placement-centric defenses across the three test IC designs.

6.2.2 Net Blockage Results. Both types of guard wires, natural and synthetic, attempt to block security-critical interconnect wires to prevent attackers from attaching rogue wires to them, thus, minimizing/eliminating possible Trojan configurations, i.e., (security-critical-wire, trigger-space)6 pairs. We use GDS2Score’s

Figure 10: Plot of the GDS2Score net blockage metric computed across three different IC layouts, with and without guard wires.

net blockage metric to compute the surface-area-coverage differences between natural and synthetic guard wires. Fig. 10 compares the net blockage computed across 9 total IC layouts: three circuit designs with no guard wires, natural guard wires, and synthetic guard wires.

Across all three designs, synthetic guard wire provide more protection than natural guard wires. Specifically, for all 3 designs, synthetic guard wires achieve 100% net blockage. This means that there is no place on any security-critical net in these designs where an attacker can attach a rogue wire. Natural guard wires are unable to achieve 100% coverage due mainly to having to meet their own routing constraints which prevents our tool from creating enough nets to block an entire security-critical wire.

6.2.3 Route Distance Results. Since DR only limits the routing resources needed to insert a Trojan at fabrication time, it is vital to understand how DR reduces the overall fabrication-time attack surface, i.e., both Trojan routing and placement resources (trigger spaces). We use GDS2Score’s route distance metric to enumerate all possible (security-critical-wire, trigger-space) pairs, i.e., all available Trojan placement and routing resources an attacker can utilize to insert a Trojan at fabrication time. Namely, we use GDS2Score’s route distance metric to illustrate the attack surface of a given IC layout.

Fig. 11 shows the route distance metric as computed across three different IC designs, with and without layout-level defenses including: 1) DR (both natural and synthetic guard wires) and 2) defensive placement. Each heatmap is intended to be analyzed column-wise, where each column is a histogram of the distances between unblocked security-critical wires and trigger-spaces within a size range. Namely, each heatmap illustrates the fabrication-time attack surface of each IC layout. If a circuit has no attack configurations, i.e., all security-critical wires are blocked or there are no trigger-spaces, the route distance heatmap is completely dark (column ratios of 0). If it is impossible to eradicate all attack configurations, the most secure layout for such a circuit would have maximum distances between unblocked security-critical wires and trigger-spaces, i.e., a heatmap with the top row the lightest color (top row ratios of 1). This is because larger distances increase the signal delay.

6*Trigger-spaces* are a GDS2Score term for contiguous open placement sites on the device layer.
for the hardware Trojan; increasing the challenge of the attacker to meet timing constraints for their attack.

Synthetic guard wires outperform natural guard wires and placement-centric defenses. For all three IC designs, synthetic guard wires were able to close the fabrication-time attack-surface by completely blocking all security-critical wires (Fig. 10). Thus, DR demonstrates an effective layout-level defense.

### 6.3 Practicality

DR is effective, but is it practical? We evaluate the cost of deploying DR across the three IC layouts. Specifically, we analyze the power, route density, and performance (timing) overheads incurred by deploying both natural and synthetic guard wires. While it is common to analyze power, performance, and area, of an IC layout, we instead analyze power, performance and route density. Area measurements of a layout refer to the device-layer area, i.e., width and length, since the height (number of routing layers) is fixed for a given process technology. Since DR does not require additional logic gates, we do not increase the width and height (area) of the core area, rather DR alters the total wire length in the design. Thus, measuring routing density overhead is more meaningful. We use the built-in features of Cadence tools to compute these overheads.

Fig. 12 details our results. Power and timing overheads were both less than 1%. In some cases, the timing was better for the guard wire
6.4 Threat Analysis

Lastly, we provide a threat analysis of DR. Recall (Fig. 5), there are three ways an attacker could bypass guard wires to carry out a fabrication-time attack: A) delete a guard wire, B) cut a guard wire, or C) re-route a guard wire. The first two attacks, deletion and cutting, are trivial to detect (§ 4). The third attack, guard wire re-route, is the most sophisticated (Fig. 5c). Only synthetic guard wires can reliably detect such an attack using TDR (Fig. 7).

6.4.1 Smallest Re-route Attack. The minimum (re-route) attack edit an adversary can make to a guard wire is to jog a top (or bottom) guard wire to an adjacent routing track, and attach to the security-critical wire from above (or below) with a via, as illustrated in Figure 5c. This edit is minimal because the the minimal metal pitch (MMP), or (horizontal) distance between the centers of adjacent routing tracks on the same layer, is much smaller than the (vertical) distance between overlapping routing tracks on adjacent routing layers. Specifically, the smallest (re-route) attack edit would increase a guard wire’s length by: \( l_{\text{attack}} = 2 \times \text{MMP}_r \), where \( \text{MMP}_r \) is the MMP on layer, \( r \), which is defined in the design rules of a given process technology. Table 1 summarizes the minimal-length-edits to a synthetic guard wire an attacker must make to bypass DR, according to the 45nm process technology we target in this study.

6.4.2 Tamper Detection with TDR. When IC interconnects are injected with a pulsed waveform with a rise time less than twice the propagation delay of the interconnect, they behave like transmission lines (Eq. (1)). Hence, time-domain reflectometry (TDR) can be used to measure several characteristics of synthetic guard wires to ensure they have not been tampered with (§ 2.4). Specifically, the lengths of each guard wire can be analyzed with TDR to detect if they have been altered. Once measured, the lengths can be compared with that predicted by a 3D electromagnetic field solver [33]. While modeling all interconnects within a large complex IC using a field solver is computationally impractical, it is practical to analyze only a small subset of interconnects, e.g., the guard wires and surrounding circuit structures [39].

Prior work demonstrates terahertz TDR systems [12, 39, 53, 55] capable of measuring the propagation delay of an interconnect to an accuracy of ±2.6 femtoseconds (fs). Such systems utilize laser-driven optoelectronic measurement techniques to achieve such high resolutions. According to the ideal transmission line model [52], the propagation delay, \( T_{pd} \), is a function of the dielectric constant, \( D_k \), speed of light, \( c \), and length of the transmission line (guard wire), \( L_{gw} \), as shown in Eq. (2).

\[
T_{pd} = \frac{\sqrt{D_k}}{C} \tag{2}
\]

Table 1: Minimum guard wire re-route attack (Fig. 5) edit–distances for each routing layer in our target (IBM 45nm SOI) process technology.

| Routing Layer | Min Wire Spacing (um) | Min Metal Pitch (um) | Min Attack Edit (um) | TDR Detectable? |
|---------------|----------------------|---------------------|---------------------|-----------------|
| 1             | 0.07                 | 0.14                | 0.28                | Yes             |
| 2             | 0.07                 | 0.14                | 0.28                | Yes             |
| 3             | 0.07                 | 0.14                | 0.28                | Yes             |
| 4             | 0.09                 | 0.19                | 0.38                | Yes             |
| 5             | 0.09                 | 0.19                | 0.38                | Yes             |
| 6             | 0.14                 | 0.28                | 0.56                | Yes             |
| 7             | 0.14                 | 0.28                | 0.56                | Yes             |
| 8             | 0.80                 | 1.60                | 3.20                | Yes             |
| 9             | 0.80                 | 1.60                | 3.20                | Yes             |
| 10            | 2.00                 | 4.00                | 8.00                | Yes             |
6.4.3 Process Variation & TDR Accuracy. TDR is the ideal tamper detection tool as process variation has no impact on its accuracy. Recall that knowing the inter-layer dielectric constant of the insulating material surrounding synthetic guard wires is all that is required to compute their lengths using TDR (Eq. (2)). The dielectric constant of the inter-layer dielectric is not dependent on its geometric properties, thus the dielectric constant is usually very well controlled [10].

6.4.4 Process Variation & Guard Wire Lengths. Detecting the minimal guard-wire re-route attack requires detecting the smallest increase to a guard wire’s length required to move it out of the way (§ 6.4.1). This distance is larger than the worst-case manufacturing process variation in a guard wire’s length. Namely, with $L_{\text{attack}}$ as the minimum length guard wire (re-route) attack edit, $L_{\text{design}}$ as the designed length of the guard wire, and $L_{\text{wc\_error}}$, as the worst-case manufacturing error in the actual guard wire’s length (or -):

$$L_{\text{design}} - L_{\text{wc\_error}} + L_{\text{attack}} > L_{\text{design}} + L_{\text{wc\_error}} \quad (3)$$

For a guard wire on routing layer $r$, the worst-case manufacturing error, $L_{\text{wc\_error}}$, can be deduced from the manufacturing design rules as:

$$L_{\text{wc\_error}} = 2 \times \frac{\text{min\_spacing}_r}{2} = \text{min\_spacing}_r \quad (4)$$

where $\text{min\_spacing}_r$ is the minimum required spacing surrounding a wire routed on metal layer $r$.

We illustrate this in Figure 13, where we plot the minimum length differences between unmodified (un-attacked) and minimally-rerouted (attacked) guard wires, overlaid with error bars indicating the worst-case range of variation in a guard wires fabricated length caused by process variation. Even in the worst case, across all routing layers, unmodified vs attacked guard wires are discernible.

6.4.5 TDR Simulation. Using the TDR propagation delay model described in Eq. 2, and the previously studied capabilities of optoelectrical terahertz TDR [12, 39, 53, 55], we simulate the difference in reflection times for single pulse TDR waveforms applied to synthetic guard wires of various lengths. Specifically we compare the reflection time of an (unmodified) 100 micron long guard wire and compare it to the reflection time observed from a similar guard wire that has been lengthened by the minimal attack edit distance for each routing layer (Tab. 1). We assume a dielectric constant of 3.9, the nominal dielectric constant of silicon dioxide [27]. Taking into account a (Gaussian) standard error (across reflection time measurements) of $\pm 2.6 \times 10^{-10}$, as reported by [39], we compute the minimum number of TDR measurements required to discriminate an unmodified guard wire from an attacked guard wire with confidence levels of 95% and 99%. We plot these results in Figure 14. Our results demonstrate that existing terahertz TDR systems are capable of detecting the minimum re-route attacks across all routing layers (Tab. 1) in our target 45nm process, requiring at most 14 and 24 TDR measurements to achieve confidence levels of 95% and 99%, respectively.

7 DISCUSSION

DR targets the prevention of fabrication-time attacks. Experiments with IC layouts of three designs show that it is effective, deployable, and tamper-resistant. Provided below is additional discussion on its signal integrity impact, flexibility, scalability, and extensibility.

Signal Integrity Impact. Routing long wires parallel to security-critical wires increases coupling capacitance, thus creating cross-talk between the guard and security-critical wires. However, synthetic guard wires are not actively driven (and can be grounded) during normal chip operation. Thus, cross-talk is not an issue—in fact, synthetic guard wires decrease cross-talk by acting as shields between security-critical signals and the rest of the circuit. Natural guard wires, on the other hand, have the potential to cause signal integrity and timing issues. These issues can be avoided by only using wires with low switching rates as natural guard wires.

Defense-in-Depth. While DR alone is capable of thwarting even the stealthiest fabrication-time attacks, its low deployment costs enable defense-in-depth. Layering DR with other preventive measures, such as Ba et al.’s defensive placement [4, 5], provides an additional layer of protection.

Scalability. Although Moore’s law is near at its limit, transistors continue to shrink. Only three companies are capable of manufacturing 7–10nm transistors [31]. It is, therefore, vital for DR to scale with process technology. With respect to deletion or cutting attacks
(Fig. 5), both natural and synthetic guard wires scale with process technology advancements as continuity checks are the only tests necessary to detect tampering. With respect to re-route attacks, synthetic guard wires will scale, provided TDR capabilities continue to scale. This is likely as advancements in microelectronic feature sizes translate to advancements in the technologies built upon them.

**Extensibility of CAD Tools.** Our DR deployment framework (§ 5) is built on top of a commercial IC CAD tool [11] and open-source VLSI analysis tools [37, 38]. Extending this framework to work across other commercial IC layout CAD tools involves incorporating support for each vendor’s CAD tool APIs. We foresee DR deployed as a deeply integrated component of commercial VLSI CAD tools as they focus more on IC security.

8 RELATED WORK

Fabrication-time attacks and defenses have been extensively studied. While most attacks have only been contrived in literature [9, 26, 28, 34, 47, 61], a recent Bloomberg business week article [45] claims to have discovered the first fabrication-time attack in the wild. While this attack was done at the PCB level, prior studies primarily focus on the IC level, where components are much smaller and hence easier to conceal. With respect to defenses, there have been many proposed techniques [1, 2, 4, 5, 7, 15, 16, 24, 32, 40, 60, 64], but besides DR, have all failed against the A2 Trojan [61].

**Fabrication-time Attacks.** Attacks have spanned the trade-space of footprint size, stealth, and controllability. Specifically, some attacks have demonstrated stealth and controllability, at the cost of large footprints [9, 26, 34], while others have demonstrated small (or non-existent) footprints, at the cost of controllability and stealth [28, 47]. The most formidable attack in particular—the A2 attack [61]—has demonstrated all three: small footprint, stealth, and controllability.

The first fabrication-time insertion of a hardware Trojan was developed by Lin et al. [34] who proposed a Trojan designed to leak information over a deliberately created side channel. Specifically, they designed and implemented a hardware Trojan, with a footprint of approximately 100 logic gates, to create an artificial power side channel for leaking cryptographic keys. Albeit unique at the time, today such a large footprint makes the attack detectable via side channel defenses [2, 7, 16].

Dopant level [9, 47] Trojans explore how altering the IC fabrication process in a targeted way can produce ICs with transistors that wear out quickly, causing the circuit to fail early. These Trojans demonstrate a fabrication-time attack where no additional circuit components need to be added, thus being extremely stealthy. However, dopant-level Trojans have several drawbacks. First, they have limited capabilities as they can only be constructed out of components already in the target IC design. Second, dopant level alterations are detectable with optical microscopy [51].

The most lethal fabrication-time attack is the A2 Trojan, developed by Yang et al. [61]. The A2 Trojan utilizes analog components to build a counter-based trigger circuit with a footprint of less than the size of one flip-flop. Its complex triggering mechanism makes it stealthy, i.e., unlikely to accidentally deploy during post-fabrication functional testing or under normal chip operation, yet is controllable from user-level software. Its unique design makes it the only Trojan to evade all detection schemes, except DR.

**Fabrication-time Defenses.** There are two fabrication-time attack defense strategies: detective or preventive. Most prior work has focused on detective strategies, while few works have focused on preventive strategies. Detective strategies involve side-channel analysis [2, 7, 24, 40], imaging [1, 64], and on-chip sensors [16, 32]. Until DR, preventive measures been placement-focused [4, 5, 60]. We highlight both detective and preventive defenses below.

The first side-channel detection scheme was proposed by Agrawal et al. [2]. They used power, temperature, and electromagnetic (EM) side-channel measurements to record a fingerprint of a “golden” IC during normal, and compared this fingerprint to one acquired from an untrusted IC. Similarly, Jin et al. [24] create a timing-based fingerprint obtained by measuring the output delays resulting from applying various input combinations to a given IC. While side-channel detection schemes are effective against hardware Trojans with large footprints, they fail at detecting Trojans like A2 [61], whose side-channel signatures are well below operational noise margins.

Like side-channel detection, imaging is another detective defense. Specifically, backside imaging is a non-destructive technique that can resolve device-layer components (Fig. 3) as this layer isn’t blocked by any wires. Zhou et al. [64] propose filling the placement grid with highly reflective fill cells, as opposed to the standard fill cells used, to encode a watermark that can be captured using backside imaging. Thus, if the watermark has been perturbed during fabrication, an attack has occurred. Unfortunately, this technique requires hours to image a single IC, provides no visibility in metal layers, and its resolution seems capped at 45 nm processes.

9 CONCLUSION

DR is the first routing-centric preventive defense against fabrication-time attacks. It makes attaching Trojan logic to security-critical logic in the victim IC infeasible by routing guard wires on all six sides of security-critical wires, eliminating locations where an adversary can attach rogue Trojan wires. DR guard wires are tamper evident, indicating if a foundry-level attacker modified them to insert a hardware Trojan. We design and implement two guard wire variants: natural and synthetic. The former consists of wires inherent to the host IC layout, that are vital to the functionality of the design, but not security-critical. The latter are not inherent to the functionality of the design, and are added during the place-and-route IC design phase. Both guard wire variants are capable of detecting deletion and cutting attacks. Using terahertz time-domain reflectometry (TDR), synthetic guard wires can also detect re-route attacks (Fig. 5) across all routing layers of our 45nm process.

We develop an automated framework for deploying both natural and synthetic guard wires. We evaluate the effectiveness, deployability, and tamper-resistance of DR across three different IC designs including a processor SoC, AES and DSP accelerators. The experimental results show that DR thwarts the insertion of even the stealthiest known additive hardware Trojans, across all 3 layouts, with power, timing, and area overheads of less than 1%.
ACKNOWLEDGMENTS

We thank Brian Tyrrell, Matt Guyton, and other members of the MIT Lincoln Laboratory community for their thoughtful feedback that enhanced the quality of our work.

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. Air Force.

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE 1256260. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Ronen Adato, Aydar Uyar, Mahmoud Zangeneh, Boyon Zhou, Ayaj Joshi, Bennett Goldberg, and M Selim Unlu. 2016. Rapid mapping of digital integrated circuit logic gates via multi-spectral backside imaging. arXiv:1605.09360 (2016).
[2] Dakhil Agrawal, Sulekh Bakshi, Deniz Karayukyula, Pankaj Rohatgi, and Berk Sunar. 2007. Trojan Detection using IC Fingerprinting. In IEEE Symposium on Security and Privacy (SP).
[3] Yousra Alkabani and Farinaz Koushanfar. 2008. DesignerÃ­s hardware Trojan horse. In IEEE International Workshop on Hardware-Oriented Security and Trust (HOST).
[4] Papa-Sidy Ba, Sophie Dupuis, Manikandan Panaliachami, Giorgio Di Natale, Bruno Rouzeyle, et al. 2016. Hardware Trust through Layout Filling: a Hardware Trojan Prevention Technique. In IEEE Computer Society Annual Symposium on VLSI (ISVLSI).
[5] Papa-Sidy Ba, Manikandan Panaliachami, Sophie Dupuis, Marie-Lise Flottes, Giorgio Di Natale, and Bruno Rouzeyle. 2015. Hardware Trojan prevention using layout-level design approach. In European Conference on Circuit Theory and Design (ECCTD).
[6] Halil B Bakoglu. 1990. Circuits, Interconnections, and Packaging for VLSI (1990).
[7] Josep Balasch, Benedikt Gierlichs, and Ingrid Verbauwhede. 2015. Electromagnetic circuit fingerprints for hardware trojan detection. In IEEE International Symposium on Electromagnetic Compatibility (EMC).
[8] Mark Beaumont, Bradley Hopkins, and Tristan Newby. 2011. Hardware trojans—prevention, detection, countermeasures (a literature review). Technical Report. Defence Science and Technology Organization Edinburgh (Australia).
[9] Georgia B. Becker, Francesco Regazzoni, Christof Paar, and Wayne P. Burleson. 2013. Stealthy dormant-level hardware trojans. In International Workshop on Cryptographic Hardware and Embedded Systems (CHES).
[10] Danone Boning and Sanj Nissoli. 2000. Models of process variations in device and interconnect. Design of high performance microprocessor circuits (2000).
[11] Cadence Design Systems [n. d.]. Innovus Implementation System. (n. d.).
[12] Yongming Cai, Zhiyong Wang, Rajen Dias, and Deepak Goyal. 2010. Electro Optical Terahertz Pulse ReflectometryAtAn innovative fault isolation tool. In Electronic Components and Technology Conference (ECTC). 2010 Proceedings 60th.
[13] Rajat Subhra Chakraborty, Seetharam Narasimhan, and Swapar Bhunia. 2009. Hardware Trojan: Threats and emerging solutions. In IEEE International High Level Design Validation and Test Workshop (HLDVT). IEEE.
[14] Ming-Kun Chen, Cheng-Chi Tai, and Yu-Jung Huang. 2006. Nondestructive analysis of interconnection in two-die BGA using TDR. IEEE Transactions on Instrumentation and Measurement (2006).
[15] Ronald P Cocchi, James P Bauskus, Lapp Wai Chow, and Bryan J Wang. 2014. Circuit camouflage integration for hardware IP protection. In Proceedings of the 51st Annual Design Automation Conference. ACM, 1–5.
[16] Domenic Forte, Chongyi Bao, and Anilur Serifatava. 2013. Temperature tracking: An innovative run-time approach for hardware Trojan detection. In IEEE/ACM International Conference on Computer-Aided Design (ICCAD).
[17] Leonard A Hayden and Vijay K Tripathi. 1994. Characterization and modeling of multiple line interconnections from time domain measurements. IEEE Transactions on Microwave Theory and Techniques (1994).
[18] Matthew Hicks, Muph Finnicum, Samuel T. King, Milo M. K. Martin, and Jonathan M. Smith. 2010. Overcoming an Untrusted Computing Base: Detecting and Removing Malicious Hardware Automatically. In Proceedings of the IEEE Symposium on Security and Privacy (SP).
[19] Matthew Hicks, Cynthia Sturton, Samuel T. King, and Jonathan M. Smith. 2015. SFECS: A Lightweight Runtime Mechanism for Protecting Software from Security-Critical Processor Bugs. In International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS).
[20] Simon Hollis and Simon W. Moore. 2006. Rasp: an area-efficient, on-chip network. In 2006 International Conference on Computer Design. IEEE, 63–69.
[21] Simon J Hollis. 2009. Pulse generation for on-chip data transmission. In 2009 12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools. IEEE, 303–310.
[22] Chung-Wen Houe and Te-Wen Pan. 1997. Reconstruction of nonuniform transmission lines from time-domain reflectometry. IEEE Transactions on Microwave Theory and Techniques (1997).
[23] Yier Jin, Nathan Kupp, and Yiorgos Makris. 2010. DFTT: Design for Trojan test. In IEEE International Conference on Electronics, Circuits, and Systems (ICECS).
[24] Yier Jin and Yiorgos Makris. 2008. Hardware Trojan detection using path delay fingerprint. In IEEE International Workshop on Hardware-Oriented Security and Trust (HOST).
[25] Shane Kelly, Xuehui Zhang, Mohammed Teheranipoor, and Andrew Ferraisnolo. 2015. Detecting hardware trojans using on-chip sensors in an ASIC design. Journal of Electronic Testing 31, 1 (2015), 11–26.
[26] Samuel T. King, Joseph Tucek, Anthony Cozzi, Chris Grier, WeiHang Jiang, and Yuanyuan Zhou. 2008. Designing and Implementing Malicious Hardware. In Proceedings of the Uesnix Workshop on Large-Scale Exploits and Emergent Threats (LEET).
[27] Angela I Kingon, Jon-Paul Maria, and SK Streffler. 2000. Alternative dielectrics to silicon dioxide for memory and logic devices. Nature (2000).
[28] Raghavan Kumar, Philipp Jovanovic, Wayne Burleson, and Ila Polian. 2014. Parametric trojans for fault-injection attacks on cryptographic hardware. In Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC).
[29] Mark Lapedus. 2017. Battling FB-Cycle Times. (February 2017). https://semengineering.com/battling-fb-cycle-times/.
[30] Mark Lapedus. 2018. Big Trouble At 3nm. (June 2018). https://semengineering.com/big-trouble-at-3nm/.
[31] Mark Lapedus. 2018. GF Puts 7nm On Hold. (August 2018). https://semengineering.com/gf-puts-7nm-on-hold/.
[32] Jie Li and John Lach. 2008. At-speed delay characterization for IC authentication and Trojan horse detection. In IEEE International Workshop on Hardware-Oriented Security and Trust (HOST).
[33] Jun Jun Lim, Nor Adila Johari, Subhash C Rustagi, and Narain D Arora. 2014. Characterization of Interconnect Process Variation in CMOS Using Electrical Measurements and Field Solver. IEEE Transactions on Electron Devices (2014).
[34] Lang Lin, Markus Kasper, Tim Güneysu, Christof Paar, and Wayne Burleson. 2009. Trojan Side-Channels: Lightweight Hardware Trojans through Side-Channel Engineering. In International Workshop on Cryptographic Hardware and Embedded Systems (CHES).
[35] Timothy Linscott, Pete Ethrel, Valeria Bertacco, and Todd Austin. 2018. SWAN: mitigating hardware trojans with design ambiguity. In 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). IEEE, 1–7.
[36] MIT Lincoln Laboratory. [n. d.]. Common Evaluation Platform. (n. d.). https://github.com/mit-lil/CEP.
[37] MIT Lincoln Laboratory. [n. d.]. GD52-Score. ([n. d.]). https://github.com/mit-ll/gd52-score.
[38] MIT Lincoln Laboratory. [n. d.]. Nemo. ([n. d.]). https://github.com/mit-ll/nemo/.
[39] Michael Nagel, Alexander Michalski, and Heinrich Kurz. 2011. Contact-free fault location and imaging with on-chip terahertz time-domain reflectometry. Optics Express (2011).
[40] Seetharam Narasimhan, Ximmi Wang, Dongdong Da, Rajat Subhra Chakraborty, and Swapar Bhunia. 2011. TeSRA: A robust temporal self-referencing approach for hardware Trojan detection. In IEEE International Symposium on Hardware-Oriented Security and Trust (HOST).
[41] C Odegard and C Lambert. 1999. Comparative TDR analysis as a packaging FA tool. In ISTFA 1999. 25th International Symposium for Testing and Failure Analysis.
[42] OpenCores.org. [n. d.]. OpenRISC OR1200 Processor. ([n. d.]). https://github.com/mit-ll/openrisc/or1200.
[43] Dan I Philen, Jan A White, Jane F Kuhl, and Stephen C Mettler. 1982. Single-mode fiber OTDR: Experiment and theory. IEEE Transactions on Microwave Theory and Techniques (1982).
[44] Misrod Potkonjak, Ali Nahapetian, Michael Nelson, and Tammara Massey. 2009. Hardware Trojan horse detection using gate-level characterization. In Proceedings of ACM/IEEE Design Automation Conference (DAC).
[45] Jordan Robertson and Michael Riley. 2018. The Big Hack: How China Used a Tiny Chip to Infiltrate U.S. Companies. (Oct 2018). https://www.bloomberg.com/news/features/2018-10-04/the-big-hack-how-china-used-a-tiny-chip-to-infiltrate-america-s-top-companies.
[46] Masoud Rostami, Farinaz Koushanfar, Jeyavijayan Rajendran, and Ramesh Karri. 2013. Hardware Security: Threat Models and Metrics. In Proceedings of the International Conference on Computer-Aided Design (ICCAD).
[47] Yuriy Shyanovskii, F Wolff, Aravind Rajendran, C Papachristou, D Weyer, and W Clay. 2010. Process reliability based trojans through NBTI and HCI effects. In NASA/ESA Conference on Adaptive Hardware and Systems (AHS).

[48] D Smolyansky. 2004. Electronic Package Fault Isolation Using TDR. ASM International (2004).

[49] PI Somalo and DL Hollway. 1969. Microwave Locating Reflectometer. Electronics Letters (1969).

[50] Ed Sperling. 2018. Design Rule Complexity Rising. (April 2018). https://semiengineering.com/design-rule-complexity-rising/.

[51] Takeshi Sugawara, Daisuke Suzuki, Ryoichi Fujii, Shigeaki Tawa, Ryohri Hori, Mitsuru Shiosaka, and Takeshi Fujino. 2014. Reversing stealthy dopant-level circuits. In International Workshop on Cryptographic Hardware and Embedded Systems (CHES).

[52] James Sutherland. 1999. As Edge speeds increase, wires become transmission lines. EDN (1999).

[53] MY Tay, L Cao, M Venkata, L Tran, W Donna, W Qiu, J Alton, PF Taday, and M Lin. 2012. Advanced fault isolation technique using electro-optical terahertz pulse reflectometry. In Physical and Failure Analysis of Integrated Circuits (IPFA), 2012 19th IEEE International Symposium on the.

[54] Mohammad Tehranipoor and Farinaz Koushanfar. 2010. A survey of hardware trojan taxonomy and detection. IEEE Design & Test of Computers 27, 1 (2010).

[55] TeraView. [n. d.]. Electro Optical Terahertz Pulse Reflectometry: The world’s fastest and most accurate fault isolation system.

[56] Mohit Tiwari, Hassan M.G. Wassel, Bita Mazloom, Shashidhar Mysore, Frederic T. Chong, and Timothy Sherwood. 2009. Complete Information Flow Tracking from the Gates Up. In International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS). 109–120.

[57] TSMC. 2019. TSMC Fabrication Schedule àŤ 2019. (April 2019). https://www.mosis.com/db/publ/ftched/ORG=TSMC.

[58] Adam Waksman, Matthew Suozzo, and Simha Sethumadhavan. 2013. FANCI: identification of stealthy malicious logic using boolean functional analysis. In Proceedings of the ACM SIGSAC Conference on Computer & Communications Security (CCS).

[59] Francis Wolff, Chris Papachristou, Swarup Bhunia, and Rajat S Chakraborty. 2008. Towards Trojan-free trusted ICs: Problem analysis and detection scheme. In Proceedings of the ACM Conference on Design, Automation and Test in Europe (DATE).

[60] Kan Xiao and Mohammed Tehranipoor. 2013. BISA: Built-in self-authentication for preventing hardware Trojan insertion. In IEEE International Symposium on Hardware-Oriented Security and Trust (HOST).

[61] Kaiyuan Yang, Matthew Hicks, Qing Dong, Todd Austin, and Dennis Sylvester. 2016. A2: Analog malicious hardware. In IEEE Symposium on Security and Privacy (SP).

[62] Rui Zhang, Natalie Stanley, Christopher Griggs, Andrew Chi, and Cynthia Sturton. 2017. Identifying Security Critical Properties for the Dynamic Verification of a Processor. In International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS).

[63] Xuehui Zhang and Mohammad Tehranipoor. 2011. RON: An on-chip ring oscillator network for hardware Trojan detection. In 2011 Design, Automation & Test in Europe. IEEE, 1–6.

[64] Boyou Zhou, Ronen Adato, Mahmoud Zangeneh, Tianyu Yang, Aydan Uyar, Bennett Goldberg, Selim Urdan, and Ajay Joshi. 2015. Detecting hardware trojans using backside optical imaging of embedded watermarks. In Proceedings of ACM/EDAC/IEEE Design Automation Conference (DAC).