ICAS: an Extensible Framework for Estimating the Susceptibility of IC Layouts to Additive Trojans

Timothy Trippel*, Kang G. Shin
Computer Science & Engineering
University of Michigan
Ann Arbor, MI
{trippel,kgshin}@umich.edu

Kevin B. Bush
Cyber Physical Systems
MIT Lincoln Laboratory
Lexington, MA
kevin.bush@ll.mit.edu

Matthew Hicks*†
Computer Science
Virginia Tech
Blacksburg, VA
mdhicks2@vt.edu

***Abstract*—The transistors used to construct Integrated Circuits (ICs) continue to shrink. While this shrinkage improves performance and density, it also reduces trust: the price to build leading-edge fabrication facilities has skyrocketed, forcing even nation states to outsource the fabrication of high-performance ICs. Outsourcing fabrication presents a security threat because the black-box nature of a fabricated IC makes comprehensive inspection infeasible. Since prior work shows the feasibility of fabrication-time attackers' evasion of existing post-fabrication defenses, IC designers must be able to protect their physical designs before handing them off to an untrusted foundry. To this end, recent work suggests methods to harden IC layouts against attack. Unfortunately, no tool exists to assess the effectiveness of the proposed defenses, thus leaving defensive gaps.**

This paper presents an extensible IC layout security analysis tool called *IC Attack Surface* (ICAS) that quantifies defensive coverage. For researchers, ICAS identifies gaps for future defenses to target, and enables the quantitative comparison of existing and future defenses. For practitioners, ICAS enables the exploration of the impact of design decisions on an IC's resilience to fabrication-time attack. ICAS takes a set of metrics that encode the challenge of inserting a hardware Trojan into an IC layout, a set of attacks that the defender cares about, and a completed IC layout and reports the number of ways an attacker can add each attack to the design. While the ideal score is zero, practically, we find that lower scores correlate with increased attacker effort.

To demonstrate ICAS' ability to reveal defensive gaps, we analyze over 60 layouts of three real-world hardware designs (a processor, AES and DSP accelerators), protected with existing defenses. We evaluate the effectiveness of each circuit–defense combination against three representative attacks from the literature. Results show that some defenses are ineffective and others, while effective at reducing the attack surface, leave 10's to 1000's of unique attack implementations that an attacker can exploit.

***Index Terms*—Hardware Security; Fabrication-time Attacks and Defenses; VLSI**

I. Introduction

The relationship between complexity and security seen in software also holds for Integrated Circuits (ICs). Since the inception of the IC, transistor sizes have continued to shrink. For example, compare the $10\,\mu m$ feature size of the original Intel 4004 processor [1] to the $10\,nm$ feature size of Intel's recently announced Ice Lake processor family [2]. Smaller transistors enable IC designers to create increasingly complex circuits with higher performance and lower power-usage. However, continuing this trend pushes the laws of physics and comes at a substantial cost: building a 3 nm fabrication facility is estimated to cost $15–20B [3].

Such costs are prohibitive for not only most semiconductor companies, but also nation states. Thus, **most hardware design houses are fabless**, i.e., while they are able to fully design and lay out an IC, they must outsource its fabrication. Outsourcing combined with the black-box nature of testing a fabricated IC requires fabless semiconductor companies to trust that their physical designs will not be altered maliciously by the foundry, also known as a *fabrication-time attack*. Previous work demonstrates several ways a fabrication-time attacker can insert a hardware Trojan into an otherwise trusted IC [4]–[6]. A2 [6] demonstrates the most stealthy and controllable IC fabrication-time attack to date, whereby a hardware Trojan with a complex, yet stealthy, analog trigger circuit is inserted into the finalized layout of a processor. Even though the inserted Trojan is small, the attacker can trigger it and escalate to a persistent software-level attack (i.e., a hardware foothold [7]) using only user-mode code.

Early work focuses on post-fabrication *detection* of hardware Trojans in ICs [8]. Broadly, there are two classes of detection: 1) side-channel analysis and 2) Trojan-activation via functional testing. Side-channel (power, timing, etc.) analysis [9]–[12] assumes that the Trojan's trigger is complex (i.e., many logic gates), and thus noticeably changes the physical characteristics of the chip. For example, inserting the large amount of extra logic required by a complex trigger into a design alters the power signature of the device. Alternatively, Trojan-activation via functional testing assumes that the Trojan's trigger is simple (i.e., few logic gates [4], [5]), and is thus easily activated by test vectors. Unfortunately, layering detection classes is *not* sufficient as it is shown possible to create an attack that is both small and stealthy [6].

To address the gaps left by post-fabrication Trojan *detection* schemes, recent work focuses on pre-fabrication, IC layout-level, Trojan *prevention* [13]–[15]. IC layout-level defenses work by:

1) increasing placement & routing resource utilization

* Work completed at MIT Lincoln Laboratory.
† Corresponding faculty author

DOI 10.1109/SP40000.2020.00083

2) increasing congestion around security-critical design components.

The lack of resources deprives the attacker of the required transistors needed to implement their Trojan trigger/attack circuits, and the increased congestion around security-critical wires acts as a barrier for the attacker attempting to integrate their Trojan into the victim design. Ideally, defenders utilize just enough resources and create enough congestion such that the attacker cannot implement and insert their attack, while keeping the design routable. Short of that, the added barriers require the attacker to expend significantly more resources (e.g., time) to insert their attack into an IC layout.[1]

Two IC layout-level defensive approaches exist: undirected and directed. **Undirected** approaches aim to (probabilistically) increase resource utilization and congestion across the *entire* layout by altering existing place-and-route parameters (e.g., core density [15]) that will likely result in increased resource utilization and congestion. More recently, a line of **directed** approaches have emerged [13], [16] that *systematically* increase utilization of *specific-regions* of the device layer, i.e., nearby security-critical components. Given that it is infeasible to occupy the entire device layer in a tamper-evident manner [13], [16], both classes of approaches may leave IC layouts vulnerable to attack by an untrusted foundry.

To identify gaps in existing defenses and guide future IC layout-level defenses, we design and implement an extensible measurement framework that estimates the susceptibility of an IC layout to foundry-level additive Trojan attacks. Our framework, *IC Attack Surface* (ICAS), estimates resilience in three dimensions that capture the essence and difficulty of inserting a hardware Trojan at an untrusted foundry:

1) **Trojan logic placement:** finding unused space to place additional circuit components
2) **Victim/Trojan integration:** attaching hardware Trojan payload to security-critical logic
3) **Intra-Trojan routing:** connecting the trigger and payload portions of the hardware Trojan

A successful attack requires all three steps.

Using ICAS, we analyze over 60 different IC layouts across three fully-functional ASIC designs: an AES accelerator, a DSP accelerator, and an OR1200 processor. For each layout, ICAS reports the coverage against four additive Trojan attacks [6], [7], [17], [18] that span the digital and analog domain as well a range of attack outcomes. ICAS's analysis reveals that all existing IC layout-level defenses are incomplete, leaving 1000's of opportunities for an attacker at an untrusted foundry to insert a hardware Trojan. An additional finding is that even though most existing countermeasures do increase the complexity of inserting a hardware Trojan, some countermeasures are ineffective. Lastly, ICAS's analysis suggests that focusing on exhausting resources on the device layer (i.e., transistors) is an incomplete defense; future defenses should also aim to increase congestion around security-critical wires.

[1]Time is the most critical resource for the attacker as IC fabrication is usually bounded in terms of turnaround time.

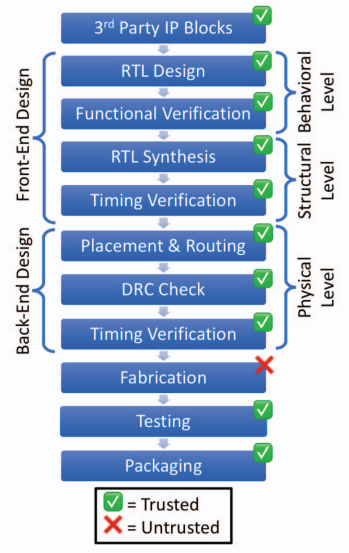

Fig. 1. The typical IC design process starts with a textual specification of design requirements and ends with a fabricated and tested chip. Green check-boxes mark trusted stages and red x-boxes mark the untrusted step (i.e., an untrusted foundry). The fabrication step takes a GDSII file (physical IC layout) as input and produces a wafer of die. While prior work proposes metrics for untrusted front-end design [17], [21]–[23], no mechanism exists for measuring an IC layout's resilience to an untrusted foundry.

This paper makes the following contributions:

- We propose an extensible methodology that estimates the difficulty of inserting additive hardware Trojans into an existing IC layout by an untrusted foundry.
- We design, implement, and open-source [19], [20] our extensible framework, ICAS, that computes various layout-specific security metrics. The ICAS framework provides an interface to programmatically query the physical layout of an IC (encoded in the GDSII format) to compute various security metrics with respect to attacks-of-interest.
- We use ICAS to estimate the effectiveness and expose the gaps of previously-proposed untrusted foundry defenses by analyzing over 60 IC layouts of three real-world hardware cores.
- We identify future directions for defenses that work in a layered fashion with existing defenses.

II. Background

A. IC Design Process

Figure 1 shows the typical IC design process [24], which consists of three main phases: 1) front-end design, 2) back-end design, and 3) fabrication. The front-end design phase can be further split into two design abstraction levels, *behavioral* and *structural*, while a single design abstraction level, *physical* (i.e., consists of both analog and digital properties), encompasses the back-end. The front-end design process begins by first describing the functionality of the circuit at the behavioral level, also known as the Register Transfer Level (RTL), using a hardware description language (HDL), like VHDL or Verilog. Next, the behavioral level description of the circuit

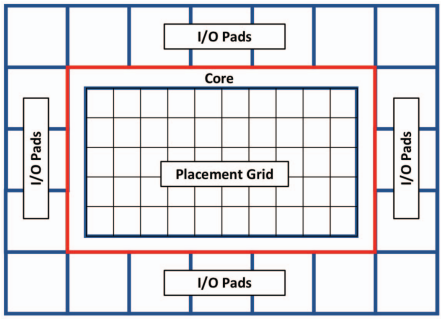

Fig. 2. Typical IC floorplan created during the place-and-route design phase. The floorplan consists of an I/O pad ring surrounding the chip core. Within the core is the placement grid. Circuit components are placed and routed within the placement grid.

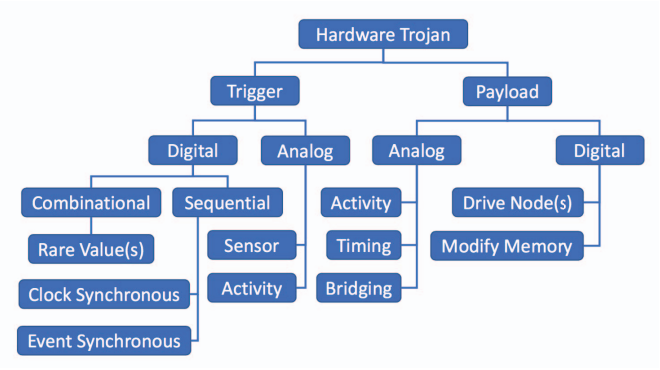

Fig. 3. An existing taxonomy of hardware Trojans [26]. This taxonomy classifies hardware Trojans based on their *trigger* and *payload* types.

is transformed into a structural level description during RTL synthesis. RTL synthesis is similar to software compilation: the RTL design is optimized and reduced to a set of logically connected digital logic gates, called a *gate-level netlist* (netlists are commonly described using an HDL language). The gate-level netlist is then passed to the back-end design phase to be transformed into something able to be implemented into a physical chip (i.e., an IC layout) through a process known as Placement and Routing (PaR).

IC layouts consist of multiple layers. The bottom layers are *device layers*, while the top layers are *metal layers*. Device layers are used for constructing circuit components (e.g., transistors), and the metal layers are used for routing (e.g., vias and wiring). The first stage of PaR is creating a floorplan. Figure 2 illustrates an IC floorplan. To create a floorplan, the dimensions of the overall chip are specified and the core area is defined. Typically a ring of I/O pads is then placed around the chip core, while a placement grid is drawn over the core. Each tile in the placement grid is known as a *placement site*. Circuit components (e.g., standard cells) are then placed on the placement grid, occupying one or more placement sites, depending on the size of the component. Lastly, all components are routed together, using one or more routing layers. The output from the back-end design is a Graphics Database System II (GDSII) file that is a geometric description of the placed-and-routed circuit layout. The GDSII file is then sent to a fabrication facility where it is manufactured. The final step is testing and packaging.

B. Hardware Trojans

*1) **Trojan Components**:* A hardware Trojan is a malicious modification to a circuit designed to modify its behavior during operation [25]. Hardware Trojans have two main components: 1) **trigger** and 2) **payload** [10], [26], [27]. Prior work classifies hardware Trojans based on the functionalities of their trigger and payload mechanisms [10], [26], [27]. In this paper, we adopt and simplify an existing hardware Trojan taxonomy [26]; shown in Figure 3.

The **trigger** mechanism of a hardware Trojan is what *initiates* the delivery of the Trojan's payload. Triggers can be built by adding, removing, or altering existing hardware in an IC. They can be digital [7] or analog [6]. The ideal trigger is *small*: requiring few or no additional circuit components, *stealthy*: requiring dozens of rare events to activate, and *controllable*: readily attacker deployable, but not so by defenders or through regular use. There have been several triggers demonstrated before that span the trade-space of large (requiring many additional gates) and stealthy [28] to the opposite: small (requiring no additional gates) and easy to trigger [4], [29]. The most advanced Trojans are small, stealthy, and controllable [6].

The **payload** mechanism receives a signal from the trigger and alters the functionality of the IC. Analog [4], [29] and digital [6] payloads exist, with a variety of effects. These effects can leak information [28], alter the internal state of the IC [6], or cause a system to be unusable (denial-of-service) [29]. Regardless of effect, the payload mechanism must route a wire to, or in the vicinity of, some target "security-critical" [30] wire in the IC design.

*2) **Trojan Implementations**:* There are three types of hardware Trojans a malicious foundry can craft into an otherwise trusted IC layout: *additive*, *substitution*, and *subtractive*. Additive Trojans involve inserting additional circuit components and/or wiring into an existing design. Substitution Trojans require removing logic with low observability to make room for additional Trojan circuit components and/or wiring in an existing circuit design. Lastly, subtractive Trojans require removing circuit components and/or wiring to alter the behavior of a existing circuit design. The focus of this paper is **estimating the susceptibility of a circuit layout to *additive* Trojan attacks**. Substitution and subtractive Trojans, while intriguing, remain largely unexplored by the community. We do not know of any demonstrably stealthy and controllable substitution or subtractive Trojans and when researchers do create such an attack, there exists orthogonal mitigation strategies [31].

Inserting an additive Trojan at an untrusted foundry requires modifying two fundamental characteristics of an IC's physical layout—placement and routing—regardless of how an attacker implements the Trojan's trigger and payload. We define ***Trojan placement*** to be the act of placing additional hardware components into an IC layout for the purpose of crafting a Trojan trigger and payload, ***Victim/Trojan integration*** to

be wiring the Trojan's payload to, or in the vicinity, of a security-critical net in the victim IC layout, and ***intra-Trojan routing*** to be the act of wiring the hardware Trojan together. The most challenging aspect of inserting a hardware Trojan at fabrication-time is finding empty space on the IC's device layer to insert the trigger and payload components (**Trojan placement**), AND routing the payload to a security-critical net (**Victim/Trojan integration**). ICAS estimates each of these fundamental tasks, in turn identifying weak points in the IC layout that an attacker might exploit.

III. Threat Model

We adopt a threat model for untrusted foundry attacks that assumes all steps in the IC design process can be trusted, *except* for all of the processes—no matter if they are outsourced—performed by a foundry (colloquially, fabrication). Figure 1 depicts our threat model. This entails that the RTL is designed, synthesized, and laid-out by trusted parties. Post fabrication testing is also performed by a trusted party. We adopt this threat model since the astronomical costs to fabricate ICs force most semiconductor companies to outsource fabrication. To this point, in 2005, the U.S. government identified the untrusted foundry threat as the most significant weakness of the microelectronics supply chain [32].

We restrict our threat model to fabrication-time attacks involving *additive* Trojans, i.e., hardware Trojans that require inserting additional circuitry into a physical IC design. Previous work on substitution/subtractive hardware Trojans shows that such Trojan insertion methods are addressable by measuring the controllability and observability of logic at the *behavioral* and/or *structural* level of the IC design, for which several methods have already been proposed [17], [21], [22], [33]–[35]. Orthogonally, this work fills the void of quantifying the susceptibility of an IC design to additive hardware Trojan insertion at the *physical* level of the IC design process by an untrusted foundry.

Focusing on additive hardware Trojans, an adversary can only insert additional components/wires. They cannot increase the size of the chip to make additional room for the implants because this is readily caught by defenders. As a result, an attacker has two choices: *find* open space in the design large enough to accommodate the additional circuitry, or *create* open space in the design by moving circuitry around. The latter is extremely challenging due to its recursive nature, it runs the risk of violating fragile timing constraints and manufacturing design rules, and it increases fabrication turnaround time (which is usually set to three months); any of which could expose the Trojan. Therefore, our focus is identifying open spaces suitable for hardware Trojan implementation.

IV. Untrusted Foundry Defenses

To protect IC layouts against insertion of a hardware Trojan by attackers at an untrusted foundry, two classes of defenses exist: **undirected** and **directed**. Undirected defenses leverage existing tuning knobs available during the IC layout process, but do not differentiate between security-critical and general-purpose wires and logic. Thus, undirected approaches provide probabilistic protection. On the other hand, directed defenses require augmenting existing PaR tool flows to harden the resulting IC layout, focusing on deploying defenses systematically around security-critical wires and logic. Thus directed approaches provide targeted protection, but increase the complexity of the place-and-route process.

This section provides an overview of the landscape of undirected and directed defenses. The focus is the mechanism each defense uses to increase the complexity faced by a foundry-level attacker. We use the results of the defensive analysis in this section to develop a set of unifying coverage metrics in the next section. Finally, in the evaluation, we evaluate commercial IC layouts using the defense-inspired metrics to quantify each defense's coverage.

A. Undirected

The lowest cost approach for protecting an IC layout from a foundry-level attacker is to take advantage of existing physical layout parameters (e.g., core density, clock frequency, and max transition time) offered by commercial CAD tools [15]. The goal is to increase congestion across the component layer and the routing layer. Ideally, this also results in increased congestion around security-critical logic and wires. Practically, increases in congestion around security-critical logic and wires is probabilistic.

Increased congestion is a symptom of increased resource utilization; hence, there are fewer resources available to the attacker. The most obvious resource that an attacker cares about are placement sites on the component layer. Increasing the density, *decreases unused placement sites*. Without sufficient placement sites, the attacker cannot implement their Trojan logic. A less obvious resource is attachment points on security-critical wires that serve as victim/Trojan integration points. Increasing routing layer congestion (via density and/or timing constraints) *increases the blockage around security-critical wires*, meaning there are less integration points.

B. Directed

To address the shortcoming of undirected approaches, recent defenses advocate focusing on security-critical logic and wires. Specifically, the approaches aim to prevent the attacker from being able to implement their hardware Trojan by occupying unused placement sites (i.e., transistors) [13], [16]. The challenge is that the filler cells used by these defenses must be tamper-evident, i.e., a defender must be able to detect if an attacker removed filler cells to implement their Trojan. Previous work shows that filling the entire component layer with tamper-evident filler cells (e.g. [15]) is infeasible due to routing congestion [16]. To make routing feasible, the most recent placement-centric defense focuses on filling the unused placement sites nearest security-critical logic first [13], [16].

Such placement-centric defenses increase the complexity faced by the attacker in two ways. First, it is harder for the attacker to find *contiguous unused placement sites* to

implement their Trojan's logic. Second, an indirect complication is increased *intra-Trojan routing* complexity. The more distributed the attacker's placement sites, the more long (i.e., uses upper routing layers) routes the attacker must create. Additionally, since the unused placement sites are far away from security critical logic, the attacker must make a longer, more complex, route to connect their hardware Trojan to the victim security-critical wire.

V. Unified Attack Metrics

Drawing from existing untrusted foundry defenses, we create an extensible set of IC layout attack metrics. We unify the objectives of existing defenses by decomposing the act of inserting a hardware Trojan into ICs at an untrusted foundry into three fundamental tasks and corresponding metrics:

1) Trojan logic placement: **Trigger Space**
2) Victim/Trojan integration: **Net Blockage**
3) Intra-Trojan routing: **Route Distance**

These tasks and accompanying metrics are the foundation for our methodology of assessing defensive coverage of an IC layout against an untrusted foundry. We implement our methodology as ICAS.

A. Challenges of Trojan Placement

The first phase of mounting a fabrication-time attack is Trojan placement. This requires locating unused placement sites on the placement grid to insert additional circuit components. While prior work [13], [15], [16] employs the notion of limiting the quantity of unused placement sites as a defense against fabrication-time attacks, how can we characterize unused placement sites to gain insight into the feasibility of a fabrication-time attack on a given IC layout?

Only 60–70% of the placement cites are occupied in a typical IC layout to allow space for routing [6]. To facilitate intra-Trojan routing, an attacker prefers open placement sites form contiguous (adjacent) regions. This allows the attacker to drop-in a pre-designed Trojan, or if one had not been pre-designed, it minimizes the intra-Trojan routing complexity by confining the intra-Trojan routing to the lowest routing layers, i.e., reducing the jumping and jogging of nets. Such adjacency is classified in image processing as "4-connected". Therefore, a key factor that determines the difficulty of mounting fabrication-time attacks is the difficulty of inserting additional circuit components into a finalized IC design. We rank this difficulty in increasing order as follows.

1) **Trivial:** the Trojan components fit within a single contiguous group of 4-connected placement sites.
2) **Difficult:** the Trojan components must be split across multiple contiguous groups of 4-connected placement sites. The more placement site groups required, the more difficult intra-Trojan routing becomes.
3) **Not Possible:** the total area required by the hardware Trojan exceeds that of available placement sites.

Figure 4 illustrates these difficulty levels. The susceptibility of an IC design to fabrication-time attack can therefore be partially quantified by the size and number of contiguous open sites on the placement grid. This is the basis for ICAS' *Trigger Space* metric.

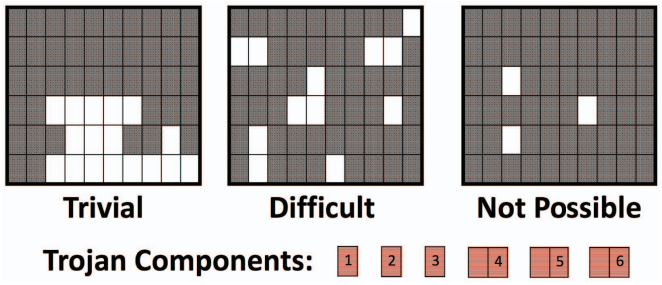

Fig. 4. Assume an attacker is attempting to insert 6 additional Trojan components that consume a total of 9 placement sites (as shown). If inserting these components on the *Trivial* placement grid (left), they can be placed adjacent to each other to simplify intra-Trojan routing. If inserting these components on the *Difficult* placement grid (middle), they must be scattered across the grid, making intra-Trojan routing more challenging. The *Not Possible* placement grid (right) does not have enough empty placement sites to accommodate the Trojan components.

B. Challenges of Victim/Trojan Integration

Routing the Trojan payload to the targeted security-critical net requires the attacker to locate the nets of interest in the IC layout. We assume the worst case: the attacker has knowledge of all security-critical nets in the design, particularly, the nets they are trying to extract information from or influence. An example of such a net in the OR1200 processor [36] is the net that holds the privilege bit. The attacker can acquire this knowledge either through a design-phase co-conspirator or through advanced reverse-engineering techniques [6]. No matter how the attacker gains this information, we assume they have it with zero additional effort.

We extend this threat to include nets that influence security-critical nets. To increase stealth, an attacker could also trace backwards from the targeted security-critical net, through logic gates, to identify nets that influence the value of the targeted security-critical net. This is called the *fan-in* of the targeted net. By connecting in this way, the attacker sacrifices controllability for stealth as their circuit modification is now physically separated from the security-critical net. To gain back controllability, attackers must create a more complex (hence larger) trigger circuit—decreasing the Trigger Space score, as well as increasing the likelihood of visual and/or side-channel detection. This trade-off limits how many levels back the attacker can integrate their payload.

No matter if the attacker is attacking the targeted security-critical wire directly or indirectly, the attacker must attach to some victim wire or route directly adjacent to it. Since an IC layout is three-dimensional, it is possible for the attacker to attach to any open point on the victim wire, either on the same layer (i.e., North, South, East, West) or from an adjacent layer (i.e., above or below). In the worst case, there are no other nets blocking the attacker from attaching to the targeted security-critical net or its N-level-deep influencers. In the best case, all attachment points are blocked by other nets.

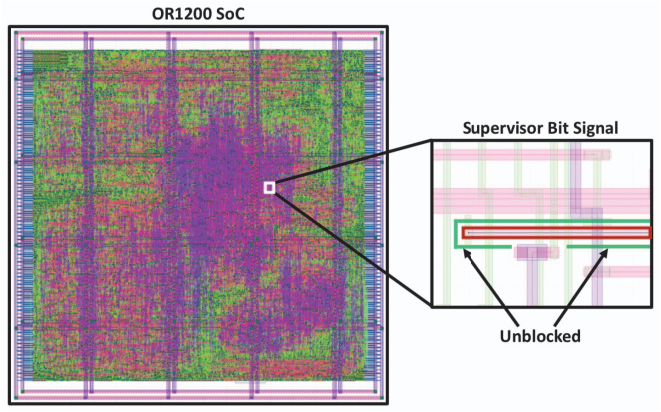

Fig. 5. The supervisor bit signal of the OR1200 processor SoC is the data input to the supervisor register of the OR1200 CPU. The supervisor register stores the privilege mode the processor is currently executing in. Changing the value on this net changes the privilege level of the processor allowing an attacker to execute privileged instructions. The more congested the area around this net, the more difficult it is for a foundry-level attacker to attach (or route in close proximity) a rogue wire to it.

To quantify the number of points along, above, and below a targeted security-critical wire—and its N-deep fan-in—we implement the *Net Blockage* metric. Figure 5 shows the open (unblocked) integration points for the privilege net on the OR1200 processor.

C. Challenges of Intra-Trojan Routing

The final phase of a fabrication-time attack is Intra-Trojan routing. Intra-Trojan routing requires connecting the components that comprise the trigger and payload portions of the hardware Trojan together—including connecting to the integration point with the victim—to form a complete hardware Trojan. In the worst case, the attacker is able to find a single contiguous region to place the trigger and payload components that is nearby the victim security-critical net. Thus, routing the trigger and payload components will be trivial and the wire used to inject the payload will be short. In the best case, the attacker will have to implement their attack using many 4-connected placement regions (i.e., low Trigger Space score) and the only integration point on the targeted security-critical net (i.e., high Net Blockage score) is as far away from the open placement regions. Hence, we focus on quantifying the difficulty of routing the payload output to open attachment points on targeted security-critical nets (and its N-deep fan-in). To this end, we identify two challenges of intra-Trojan routing:

- Comply with design and fabrication rules
- Meet Trojan and payload-delivery timing requirements

*1) **Complying with Design Rules**:* For each process technology, there are many rules associated with how wires and components must be laid out in a design. Some of these rules are defined in the Library Exchange Format (LEF) [37] and contained in files that are loaded by modern Computer Aided Design (CAD) tools throughout the IC design process. There are two types of design rules: 1) those regarding the construction of circuit components (i.e., standard cells), and 2) those regarding routing. We classify these as *component design rules* and *routing design rules*, respectively. As technology nodes shrink, both rule sets are becoming increasingly complex [38].

It is vital for an attacker to comply with these design rules as violating them risks exposure. If an attacker inserts additional logic gates (standard cells) by making copies of existing components in a design, they can avoid violating *component design rules* involved with Trojan placement. However, to connect a wire from the Trojan payload to security-critical target net(s), they must perform custom Trojan routing. Therefore, complying with *routing design rules* is a concern. Routing design rules include specifications for the minimum distance between two nets on a specific routing layer, the minimum width of nets on a given layer, etc. Complying with these rules becomes easier for an attacker if security-critical net(s) are not blocked by other wires or components. The higher the Net Blockage score, the more difficult it is to make a connection, the more complex—and error prone—the route.

*2) **Meeting Timing Requirements**:* Every wire in an IC has a resistance and a capacitance, making it behave like an RC circuit, i.e., there is a time delay associated with driving the wire $high$ (logic 1) or low (logic 0). The longer the wire, the more time delay there is [39]. If the security-critical net(s) has timing constraints (e.g., setup and hold times) that dictate when the payload signal must arrive for the attack to be successful, the Trojan routing must meet these constraints. Furthermore, the farther the security-critical net is from the payload circuit, the more obstacles that must be routed around, increasing the routing distance even further. This is the basis for ICAS' *Route Distance* metric. A natural limit for Route Distance is dictated by the clock frequency of the victim circuit, as most attacks must operate synchronously with their victim.

VI. Extensible Coverage Assessment Framework

The ICAS framework is comprised of two tools, **Nemo** and **GDSII-Score**, as shown in Figure 6. Nemo identifies security-critical wires based on designer annotations and circuit dataflow, while GDSII-Score assesses the defensive coverage of a given IC layout against a set of attacks. ICAS takes as input four sets of files: 1) gate-level netlist (generated *after* all physical layout optimizations), 2) process technology files, 3) physical layout files, and 4) set of attacks. The process technology files include a Library Exchange Format (LEF) file and layer map file [37], [40]. The physical layout files include a Design Exchange Format (DEF) file and the GDSII file of an IC layout [37], [41]. The attack files are are a list of properties for each attack to assess coverage against: number of transistors, security-critical wire(s) to attach to, and timing constraints. All ICAS input files except the attack files are either generated-by or inputs-to the back-end IC design phase, and hence are readily available to back-end designers.

Though ICAS is extensible, our implementation includes three security metrics that capture the challenges faced by a foundry-level attacker looking to insert a hardware Trojan: amount and size of open-placement regions (Trigger Space),

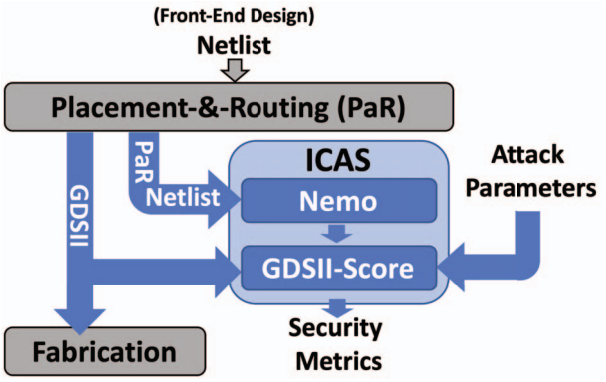

Fig. 6. ICAS consists of two tools, **Nemo** and **GDSII-Score**, and fits into the existing IC design process (Fig. 1) between PaR and fabrication. Nemo analyzes a gate-level (PaR) netlist and traces the fan-in to security-critical nets in a design. GDSII-Score analyzes a GDSII file (i.e., an IC layout) and computes metrics quantifying its vulnerability to a set of foundry-level attacks.

quantity of viable attachment points to targeted security-critical (and influencer) nets (Net Blockage), and the proximity of open placement regions to targeted security-critical net(s) (Route Distance). Together with the attack requirements, these metrics quantify the complexity an attacker faces for each step of inserting specific hardware Trojans into the given IC layout. We describe the implementation of both ICAS components below.

A. Nemo

Nemo is the first analysis tool in the ICAS framework. It bridges the semantic gap between the human readable RTL netlist and post-PaR netlist. Additionally, Nemo broadens the set of "security-critical" nets by performing a fan-in analysis of root security-critical nets. This is necessary since the interconnected nature of signals within a circuit design means an adversary could influence the state of security-critical nets by controlling a net that is a part of its fan-in. Nemo takes as input a Verilog netlist and automatically identifies security-critical nets in the post-PaR netlist HDL, which it outputs in the form of a Graphviz dot file. Similar to prior work [42]–[44], Nemo assumes that a unique signal name prefix (within the RTL HDL) has been appended to various signals considered "security-critical". We make this assumption since determining what signals are "security critical" requires contextual knowledge of how the design will be used.

1) ***Annotating Security-Critical Signals in the RTL Netlist****:* The process of uncovering and annotating security-critical signals in the RTL netlist is Security-Critical Component Identification (SCCI). While SCCI is an active area of research in the hardware security community, orthogonal to addressing the untrusted foundry problem, there are two approaches we are aware of: *manual* and *semi-autonomous* identification. The first, and most traditional, is *manual identification*. Manual identification requires a human expert to study the design's specification (e.g., Instruction Set Architecture in the case of a processor), and identify properties that are critical to the security of software or other hardware utilizing the design [30], [42]. The second, and current state-of-the-art developed by Zhang *et al.* [44], is *semi-autonomous identification*. *Semi-autonomous identification* involves two steps. First, a program observes a variety of test-benches exercising the design to generate a large set of possible invariants defined over the hardware specification. Second, a pre-trained penalized logistic regression classifier is used to classify which invariants, or portions of the specification, are security-critical. This method of SCCI is *semi-autonomous*, as it requires the classifier model be pre-trained with either existing published errata on previous versions of the hardware design, or using *manual identification*. While we perform manual SCCI, results reported by Zhang *et al.* [44] suggest that their tool would result in a similar set of root security-critical signals.

2) ***Identifying Security-Critical Signals in the PaR Netlist****:* While there are existing (aforementioned) techniques for identifying and annotating security-critical components in the RTL netlist, unfortunately, these techniques do not track security-critical signals past the RTL design phase and do not capture data-flow. Thus, Nemo's core task is to bridge the semantic gap and uncover duplicated or renamed security-critical signals in the post-PaR netlist. Fortunately, while synthesis and layout tools do modify a netlist by duplicating and removing signals and components (as part of optimization and meeting performance requirements), they do not completely rename existing signals. This makes it possible for Nemo to identify root security-critical signals (flagged at the behavioral level) by name at the physical level. ***To avoid removal of security-critical signals, we modify synthesis and layout scripts to essentially lock them in place.*** Nemo works backwards from root security-critical signals to identify the fan-in to these signals. The search depth is a configurable parameter of Nemo.

3) ***Implementation****:* Nemo is implemented as a back-end target module to the open-source Icarus Verilog (IVL) [45] Verilog compiler and simulation tool written in C++. The IVL front-end exposes an API to allow third-parties to develop custom back-end target modules. Nemo is a custom target module (also written in C++) designed to be loaded by IVL. Since gate-level netlists are often described with the same HDL that was synthesized to generate the netlist (e.g., Verilog), we utilize the IVL front-end to interpret the Verilog representation of the netlist and our custom back-end target module, Nemo, to perform a breadth-first search of the post-PaR netlist. We open-source Nemo [19] and release instructions on how to compile and integrate Nemo with IVL.

B. GDSII-Score

GDSII-Score is the second analysis tool in the ICAS framework. GDSII-Score is an extensible Python framework for computing security metrics of a physical IC layout. It takes as input the following: Nemo output, GDSII file, DEF file, technology files (LEF and layer-map files), and attacks description file. First, GDSII-Score loads all input files and locates the security-critical nets within the physical layout. Next, it computes security metrics characterizing the suscepti-

bility of an IC design to each of the input attacks. Specifically, the three security metrics that we implement are: 1) **Trigger Space**: the difficulty of implementing the hardware Trojan, 2) **Net Blockage**: the difficulty of Trojan/victim integration, and 3) **Route Distance**: the difficulty of meeting Trojan timing constraints. We open source the GDSII-Score framework and our security metric implementations [20].

*1) **Metric 1: Trigger Space**:* The Trigger Space metric estimates the challenges of Trojan placement (§V-A). It computes a histogram of open 4-connected regions of all sizes on an IC's placement grid. The more large 4-connected open placement regions available, the easier it is for an attacker to locate a space to insert additional Trojan circuit components at fabrication time. A placement site is considered to be "open" if the site is empty, or if it is occupied by a filler cell. Filler cells, or capacitor cells, are inserted into empty spaces during the last phase of layout to aid fabrication. Since they are inactive, an attacker can create empty placement sites by removing them, without altering the functionality or timing characteristics of the victim IC.

To compute the trigger space histogram, GDSII-Score first constructs a bitmap representing the placement grid. Placement sites occupied by standard cells (e.g., NAND gate transistors) are colored while those that are open are not. Information about the size of the placement grid and the occupancy of each site in the grid is available in the Design Exchange Format (DEF) file produced by commercial PaR tools. GDSII-Score then employs a breadth-first search algorithm to enumerate the maximum size of all 4-connected open placement regions.

*2) **Metric 2: Net Blockage**:* The Net Blockage metric estimates the challenges of integrating the hardware Trojan's payload into the victim circuit (§V-B). It computes the percent blockage around security-critical nets and their influencers. The more congested the area surrounding security-critical nets, the more difficult it is to attach the Trojan circuitry to these nets. There are two types of net blockage that are calculated for each security-critical net: *same-layer* and *adjacent-layer*.

Same-layer blockage is computed by traversing points around the perimeter (North, South, East, West) at a granularity of g, at a specific distance, d, around the security-critical net and determining which points lie within other circuit components, as detailed in Figure 7a. To determine if a specific point along the perimeter lies within the bounds of another circuit component, we utilize the point-in-polygon ray-casting algorithm [46]. The extension distance, d, around the security-critical path element and the granularity of the perimeter traversal, g, are configurable in our implementation. However, we default to an extension distance of one wire-pitch and a granularity of 1 database units, respectively, as defined in the process technology's LEF file. The IC designs used in our evaluation are built using a 45 nm process technology, for which 1 database units is equivalent to 0.5 nm. Additionally, an open region is considered "blocked" if it is not wide enough for a minimal width wire to be routed through while maintaining the minimal amount of wire spacing required on that metal layer, as defined in the LEF file. The percentage of the perimeter length that is blocked by other circuit components is considered the same-layer blockage percentage.

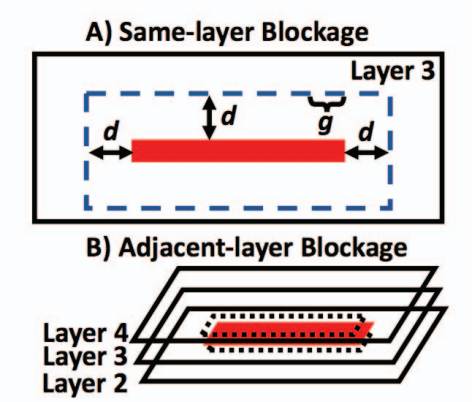

Fig. 7. A) Same-layer net blockage is computed by traversing the perimeter of the security-critical net, with granularity g, and extension distance d, and determining if such points lie inside another component in the layout. B) Adjacent-layer net blockage is computed by projecting the area of the security-critical net to the layers above and below and determining the area of the projections that are occupied by other components.

Adjacent-layer blockage is computed by analyzing the area directly above and below a security-critical net, and computing the total area of overlap between other components, as detailed in Figure 7b. To calculate this overlap area we utilize an overlapping sliding window approach. Additionally, any un-blocked regions above or below the security-critical net are considered "blocked" if they are not large enough to accommodate the smallest possible via geometry allowed on the respective via layer, as defined in the LEF file. The percentage of the total top and bottom area that is blocked by nearby circuit components is the adjacent-layer blockage percentage.

The same-layer and adjacent-layer blockage percentages are combined via a weighted average to form a comprehensive *overall* net blockage percentage where 66% is based on same-layer blockage (north, south, east, and west) and 33% is based on adjacent-layer blockage (top and bottom). We weight the same-layer blockage by 66%, or $\frac{2}{3}$, because 4 out of 6 total sides of a wire (**north**, **south**, **east**, **west**, top, and bottom) are on the same layer. Likewise, we weight the adjacent-layer blockage by 33%, or $\frac{1}{3}$.

Lastly, a total *same-layer*, *adjacent-layer*, and *overall* net blockage metric is computed for the entire IC design. For an IC design with n security-critical nets, the *same-layer* (b_{same}), *adjacent-layer* (b_{adjacent}), and *overall* (b_{overall}) net blockage metrics are computed according to equations 1, 2, and 3, respectively.

$$b_{\text{same}} = \frac{\sum_{i=1}^{n} perimeter\_blocked_n}{\sum_{i=1}^{n} perimeter_n} \tag{1}$$

$$b_{\text{adjacent}} = \frac{\sum_{i=1}^{n} area\_blocked_n}{\sum_{i=1}^{n} 2 * area_n} \tag{2}$$

$$b_{\text{overall}} = \left(\frac{2}{3} * b_{\text{same}}\right) + \left(\frac{1}{3} * b_{\text{adjacent}}\right) \tag{3}$$

TABLE I
HARDWARE TROJANS USED IN DEFENSIVE COVERAGE ASSESSMENT.

Trojan	# Std Cells	# Placement Sites	Timing Critical?
A2 Analog [6]	2	20	✗
A2 Digital [6]	91	1444	✓
Privilege Escalation [7], [17]	25	342	✓
Key Leak [18]	187	2553	✓

*3) **Metric 3: Route Distance**:* The Route Distance metric combines the Net blockage and Trigger Space metrics (thus is correlated with these metrics) to estimate the difficulty of of meeting Trojan and attack timing constraints (§V-C). It computes a conservative estimate, i.e., Manhattan distance, for the minimal routing distance between open trigger placement sites and the n least blocked integration sites on the targeted security critical nets. It cross-references each Manhattan distance with the distribution of net lengths within the entire IC design. Net length can impact whether or not the Trojan circuit will meet timing constraints and function properly. Understanding where in the distribution of net lengths the Trojan routing falls provides insights into the ability of the Trojan circuit to meet its timing requirements and is an opportunity for outlier-based defenses. In summary, the more Manhattan distances that fall within one standard deviation of the mean net length, the easier it is to carry out an attack.

We implement the Route Distance metric as follows. First, the Net Blockage and Trigger Space metrics are computed. Next, the the distribution of all net-lengths within the IC layout are computed. Then, two-dimensional Manhattan distances between all unblocked nets ($< 100\%$ overall net blockage) and trigger spaces are calculated. The Manhattan distance calculated is the minimum distance between a given trigger space and security-critical net, i.e., the minimum distance between any placement site within the given trigger space and any unblocked location on the targeted security-critical net. Lastly, each Manhattan distance is reported in terms of standard deviations away from the mean net-length in the given IC layout.

VII. EVALUATION

We use ICAS to quantify the defensive coverage of existing defensive layout techniques—revealing that gaps persist. First, we analyze the effectiveness of undirected defenses [15]. Specifically, we measure the impact of varying both physical and electrical back-end design parameters of the same IC layout on its susceptibility to attack. Second, we analyze the effectiveness of directed defenses [13], [16]. Specifically, we measure the coverage of existing, placement-oriented, defensive layout schemes in preventing the insertion of an attack by the foundry. Beyond revealing gaps, our results reveal that there is an opportunity for improving both directed and undirected defenses that systematically eliminates Trojan/victim integration points. Lastly, our evaluation also demonstrates that ICAS is design-agnostic, works with commercial tools, and scales to complex IC layouts.

A. Experimental Setup

We utilize three IC designs for our evaluations: *OR1200 processor SoC*, *AES accelerator*, and *DSP accelerator*. The OR1200 processor SoC is an open-source design [36] used in previous fabrication-time attack studies [6]. The AES and DSP accelerator designs are open-sourced under the Common Evaluation Platform (CEP) benchmark suite [47]. The OR1200 processor SoC consists of a 5-stage pipelined OR1200 CPU that implements the 32-bit OR1K instruction set and Wishbone bus interface. The AES accelerator supports 128-bit key sizes. The DSP accelerator implements a Fast Fourier Transform (FFT) algorithm.

All designs target a $45nm$ Silicon-On-Insulator (SOI) process technology. We synthesize and place-and-route all designs with Cadence Genus (version 16.23) and Innovus (version 17.1), respectively. In our first evaluation (§VII-B) the design constraints (clock frequency, max transition time, core density) used for both synthesis and layout are varied as noted. However, in our second evaluation (§VII-C) the same design constraints (100 MHz clock frequency, 100 ps max transition time, 60% core density) were used for both synthesis and layout to form a common baseline. 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 (version 6.9).

*1) **Security-critical Signals**:* The first tool in the ICAS flow is Nemo. Nemo tracks security-critical signals from the HDL level to the IC layout level. The first step is flagging root security-critical signals at the RTL level, for each IC design. For the OR1200 processor SoC, the supervisor bit signal $supv$ is flagged. We select this signal because one can alter the state of this bit to escalate the privilege mode of the processor [6]. For the AES accelerator, we flag all 128 key bits as security-critical. The $next\_out$ signal within the DSP accelerator was flagged as security-critical. The $next\_out$ signal of the DSP accelerator 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 DSP accelerator. Lastly, Nemo marks, for each design's IC layout, all root security-critical nets and their 2-deep fan-in as security-critical nets.

*2) **Hardware Trojans**:* Table I lists the hardware Trojan designs that we use in our evaluation. The first two Trojan designs (analog and digital variants of A2) are attacks on the OR1200 processor and DSP accelerator ICs. With respect to the OR1200, the A2 attacks act as a hardware foothold [7] for a software-level privilege escalation attack. With respect to the DSP accelerator, the A2 attacks suppress the *next_out* signal (§VII-A). The Privilege Escalation Trojan targets solely the OR1200 and the Key Leak solely the AES accelerator.

*3) **Build Environment**:* Both ICAS tools (Nemo and GDSII-Score) were run on the same server as the synthesis and place-and-route CAD tools. Nemo and Icarus Verilog were compiled from source using GCC (version 4.4.7). For increased performance, GDSII-Score was executed using the PyPy Python interpreter with JIT compiler (version 4.0.1).

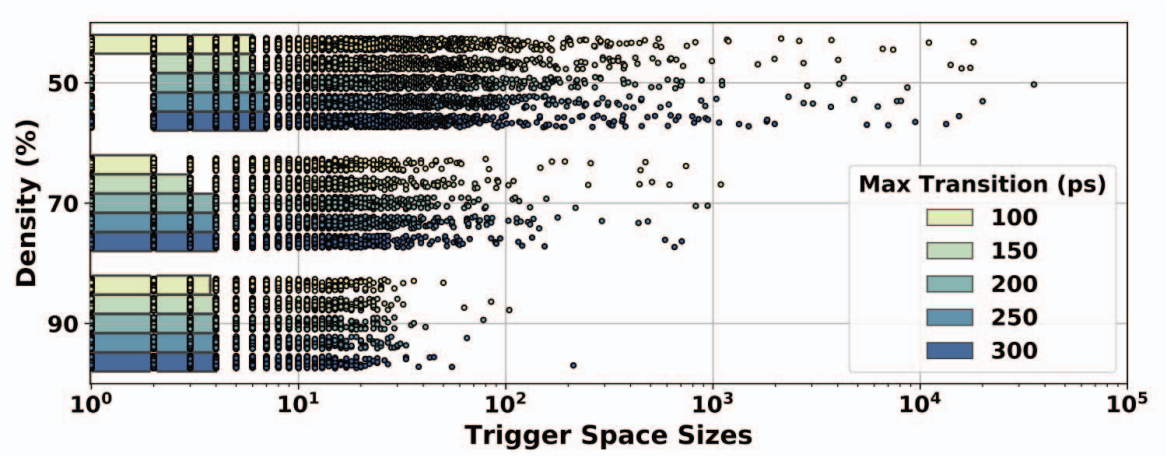

Fig. 8. Trigger Space distributions for 15 different OR1200 processor IC layouts. Core density and max transition time parameters are varied across the layouts, while target clock frequency is held constant at 1 GHz. The boxes represent the middle 50% (interquartile range or IQR) of open placement regions in a given layout, while the dots represent individual open placement region sizes.

B. Undirected Defense Coverage

As detailed in §IV-A, a defensive strategy for protecting an IC layout from foundry-level attackers is to exploit physical layout parameters (e.g., core density, clock frequency, and max transition time) offered by commercial CAD tools to increase congestion—hopefully around security-critical wires. The tradeoff is that while this is a low cost defense in that CAD tools already expose such knobs, the entire design is impacted and there is no guarantee that security-critical wires will be protected. We use ICAS and its three security metrics to quantify the effectiveness of such undirected approaches [15].

To uncover the impact of each parameter, we start by generating 60 different physical layouts of the OR1200 processor design by varying:

1) **Target Core Density** (%): 50, 70, 90
2) **Clock Frequency** (MHz): 100, 200, 500, 1000
3) **Max Transition Time** (ps): 100, 150, 200, 250, 300

Target core density is a measure of how congested the placement grid is. Typically, designers select die dimensions that achieve ∼60–70% placement density to allow space for routing [6]. Target clock frequency is the desired speed at which the circuitry should perform. Typically, designers select the clock frequency based on performance goals. Max transition time is the longest time required for the driving pin of a net to change logical values. Typically, designers choose a value for max transition time based upon power consumption and combinational logic delay constraints.

For each of the 60 layout variations we compute ICAS metrics. Figures 8, 9, and 10 provide a visual representation for each metric. Overlaid on Figure 10 are the number of unique attack (color-coded) implementations for each Trojan (Tab. I) at six parameter configurations. Across the 60 IC layouts, the time it took ICAS to complete its analyses ranged from 38 seconds to 18 minutes. On average, this translates to less than 10% of the combined synthesize and place-and-route runtimes. These run-time results demonstrate the deployability of ICAS as a back-end design analysis tool. Overall, our evaluation indicates that while some of these layout parameters do increase attacker complexity, none are sufficient on their own. We break down the results metric-by-metric.

*1) **Trigger Space Analysis**:* Figure 8 shows the distributions of open trigger spaces across 15 unique OR1200 layouts. We vary target core density and max transition time parameters across layouts, while we fix the target clock frequency at 1 GHz. A *trigger space* is defined as a contiguous region of open placement sites on the device layer placement grid and is measured by number of contiguous "4-connected" placement sites. Each box represents the middle 50%, or interquartile range (IQR), of open trigger space sizes for a given IC layout. The dots represent individual data points within and outside the IQR. Our empirical results affirm prior notions [13], [15], [16] that increasing the target core density of an IC layout results in fewer large open spaces to insert hardware Trojans. Additionally our results indicate that at lower densities, decreasing the max transition time constraint decreases the median trigger space size. Similar trends occur at lower clock frequencies. While results show that modulating target core density is effective, observe that even in the best case, large trigger spaces remain.

From our Trigger Space analysis, we conclude future undirected defenses should modulate layout parameters that both 1) shrink the trigger space IQR, and 2) shift the median towards one. In doing so, defenders: 1) minimize the variation in sizes of contiguous open-spaces available to the attacker—therefore limiting their Trojan design (size) options, and 2) force the attacker to have to distribute Trojan components across the die making *Trojan logic placement* and *intra-Trojan routing* more challenging.

*2) **Net Blockage Analysis**:* Figure 9 shows the Net Blockage metric (Eq. 3) computed across 20 unique OR1200 layouts. We fix the target density at 50% across all layouts, while the target clock frequency and max transition time are varied (as listed above). The results show that at lower clock frequencies a smaller max transition time parameter corresponds to increased Net Blockage. This corresponds to

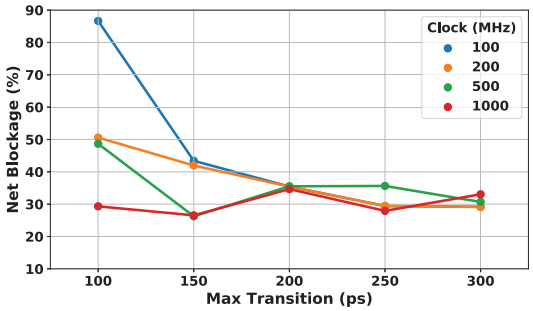

Fig. 9. *Overall* Net Blockage results computed across 20 different OR1200 processor IC layouts. A target density of 50% was used for all layouts, while target clock frequency and max transition time parameters were varied.

less open Trojan/victim integration points available to the attacker. However, as clock speed increases, the correlation between max transition time and overall Net Blockage deteriorates. Intuitively, smaller max transition times should lead to smaller average net-lengths within the design, as transition time is a function of the capacitive load on the net's driving pin [39]. Shorter net-lengths result in more routing congestion as components cannot be spread-out across the die. However, capacitive load (on a driving pin) is inversely proportional to frequency, thus at higher clock frequencies the max-transition time constraint is more easily satisfied, and altering it has less effect on the Net Blockage. Given these results, the effectiveness of modulating transition time is context dependent and—even in the best case—open integration points remain.

From our Net Blockage analysis, we conclude future undirected defenses should modulate layout parameters that both 1) shrink overall security-critical wire lengths, and 2) maximize routing congestion in the vicinity of security-critical wires. In doing so, defenders minimize the *Victim/Trojan integration* attack surface.

3) ***Route Distance Analysis****:* Figure 10 shows the Route Distances across six various OR1200 layouts in the form of heatmaps that capture the trade space between layout parameters. Core density and max transition times were varied across the layouts (indicated in the labels), while clock frequency was held constant at $100\,MHz$. Each heatmap describes several (column-wise) histograms of Route Distances in terms of standard deviations from the mean net length observed in that particular IC layout (y-axis). The Route Distances reported are those between any unblocked security-critical nets, and trigger spaces large enough to hold an attack of a given size range (x-axis). That is, the color intensities within in a given heatmap column indicate the percentage of (security-critical-net, trigger-space) pairs in that column that are within a range of distance apart. Additionally, overlaid on each heatmap are rectangles indicating the region of the heatmap where a given attack (Tab. I) can be implemented, and the number of possible attack configurations, (security-critical-net, trigger-space) pairs, that can be exploited.

If timing is critical to the operation of an attacker's desired Trojan, (critical-net, trigger-space) pairs with routing distances significantly greater than the average net length in the IC

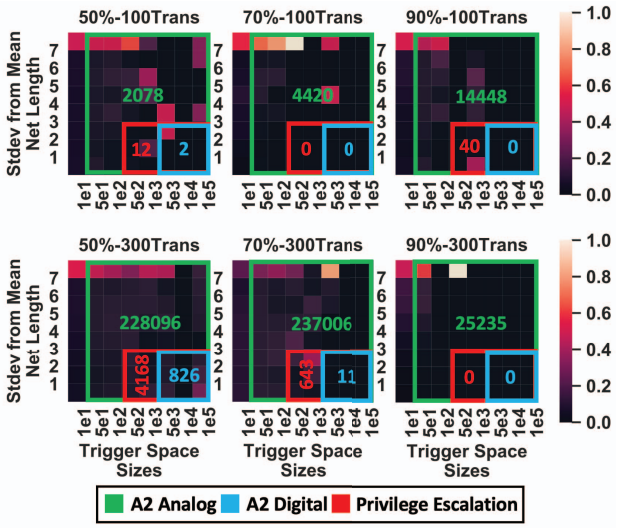

Fig. 10. Heatmaps of routing distances across six unique IC layouts of the OR1200 processor. Core density and max transition times are labeled. Each heatmap is to be read column-wise, where each column is a histogram, i.e, the color intensity within a heatmap column indicates the percentage of (critical-net, trigger-space) pairs that are within a (y-axis) distance apart. Overlaid are rectangles, indicating regions on each heatmap a given attack can exploit, and numbers indicating the number of unique attack implementations.

layout are less likely to be viable candidates for constructing hardware Trojans. IC layouts with few desirable (critical-net, trigger-space) pairs are much more time-consuming to attack. Namely, the IC layouts with heatmaps that indicate a higher percentages of far-apart (critical-net, trigger-space) pairs, where the trigger spaces are small, are most secure. From Figure 10, we conclude that at high density, max transition time has little affect on IC layout security; while at lower densities, lower max transition time designs are more secure. Similar trends exist across other layout parameters, as shown in Figures 13–15 in Appendix A.

From our Route Distance analysis, we conclude future undirected defenses should modulate layout parameters that maximimze the distance between security critical wires and open trigger spaces. In doing so, defenders: 1) maximize intra-Trojan routing difficulty, and 2) restrict attackers from implanting timing-critical Trojans.

4) ***Cost of Varying Layout Parameters****:* The results indicate that increasing core density is effective, but incomplete, and increasing clock frequency and decreasing max transition time is marginally effective and incomplete. While tuning these parameters is low cost to the designer, there is a cost to the design in terms of complexity and power requirements. We elucidate by discussing how varying each design parameter (density, clock frequency, and max transition time) impacts non-security characteristics of a circuit design.

While increasing core density to 90% makes placing-and-routing a Trojan more difficult, it also makes placing-and-routing the rest of the design more challenging. Specifically, it can become nearly impossible to meet timing closure for the entire design if there is not enough space within the core area to re-size cells and/or add additional buffer cells. Depending

on performance and security requirements, a layout engineer may choose to relax timing constraints in order to achieve a higher core density. Alternatively, a layout engineer may attempt to surround security-critical nets with areas of high densities, while maintaining a lower overall core density, as previously suggested [13], [16].

Decreasing the maximum transition time and increasing the clock speed of an entire circuit design makes it more difficult to place-and-route a functional Trojan that meets timing constraints, but also directly impacts the performance characteristics of the circuit. Additionally, it is important to note that max transition time is related to the clock frequency, so varying one without the other changes performance tolerances. While increasing the performance of the design might increase security, it comes at the cost of increasing power consumption. Depending on the power-consumption requirements of the design, it may be possible for a designer to over-constrain these parameters for added security.

C. Directed Defense Coverage

As an alternative to probabilistically adding impediments to the attacker inserting a hardware Trojan, recent works proposes a directed approach. As detailed in §IV-B, placement-centric directed defenses [13], [16] attempt to prevent the attacker from implementing their Trojan by occupying all open placement sites with tamper-evident filler cells. The limitation with such defenses is that it is infeasible to fill *all* open placement sites with tamper-evident logic [16]. Thus, the defenses focus their filling near security-critical logic, leaving gaps near the periphery of the IC layout. Whether these open placement sites near the periphery are sufficient to implement an attack is an open question.

The goal of this evaluation is to determine not only if it is still possible for a foundry-level attacker to insert a hardware Trojan, given placement-centric defenses, but to quantify the number of viable implementations available to the attacker—to act as a surrogate for attacker complexity. For the evaluation, We use our three IC designs (OR1200 processor SoC, AES accelerator, and DSP accelerator). For each design, we create two IC layouts: (1) unprotected and (2) protected. For the protected IC layout, we use the latest placement-centric defense [13]; using the identified security-critical wires (§VII-A) to direct the defense. We lay out all IC designs using these parameters: target clock frequency of $100\,MHz$, max transition time of $100\,ps$, and a target core density of 60%.

We then use ICAS to asses the defensive coverage of each of the six IC layouts. This analysis has two goals: (1) determine whether the IC is vulnerable to attack and (2) understand the impact of applying the defense. We answer both questions in an attack-centric manner using the hardware Trojans in Table I to asses defensive coverage against. For each attack/IC layout combination we plot the number of (security-critical-net, trigger-space) pairs that could be used in implementing each Trojan. A (security-critical-net, trigger-space) pair is considered a viable candidate for implementing a Trojan if:

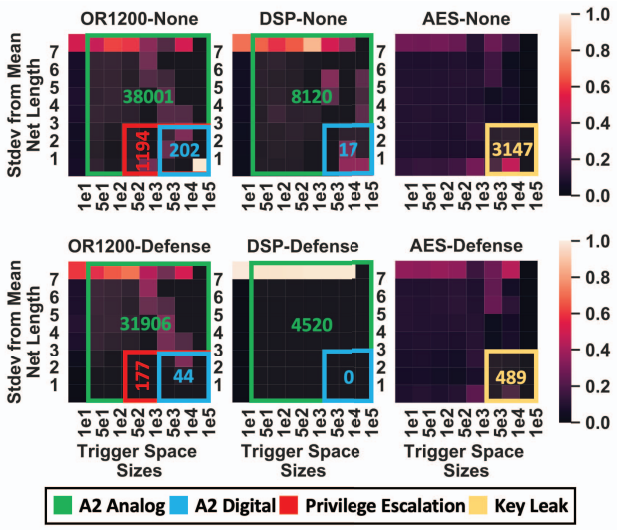

Fig. 11. Routing Distance heatmaps across three IC designs, with and without the placement-centric defense described in [13], [16]. Heatmaps should be interpreted similar to Fig. 10.

1) the trigger space size is at *least as large* as the minimum number of placement sites required to implement the desired hardware Trojan design
2) the security-critical net is less than 100% blocked
3) if the hardware Trojan is "Timing-Critical", i.e., it must function at the design's core operating frequency, then the distance between the trigger space and open integration point on the security-critical net must be ≤ 3 standard deviations from average net length; otherwise, any distance is allowed.[2]

Figure 11 shows the defensive coverage for each IC design. Overlaid on each heatmap are rectangles (and numbers) indicating unique possible attack implementations. These results show that existing placement-centric defenses are effective at *reducing* an IC's fabrication-time attack surface, compared to no defense—but *gaps persist*. Given that filling placement sites with tamper-evident logic is already maximized, these results point to systematically adding congestion around security-critical wires as a means to close all remaining defensive gaps; i.e., a directed version with similar effect to existing undirected defenses.

VIII. Discussion

ICAS is the first tool to provide insights into the security of physical IC layouts. It is extensible across many dimensions including CAD tools, process technologies, security metrics, and fabrication-time attacks and defenses. To demonstrate ICAS' capabilities we implemented three security metrics (net blockage, trigger space, and routing distance) using it. The focus of this paper is using these metrics to estimate the coverage of existing untrusted foundry defenses, which show

[2]Three standard deviations from the average net length is the threshold for Trojan-to-integration-point routing without violating timing constraints, because it accounts for 99.7% of the designs' wires—outliers tend to be power wires. For an exact calculation, it is possible to extract parasitics for a target Trojan's route to determine if it violates timing constraints.

that IC designs are still vulnerable to attack. We envision uses for ICAS beyond this, as an integral part of the IC design process using commercial tools.

*1) **ICAS-Driven Defensive Layout**:* ICAS provides an added notion of security to the IC layout (place-and-route) process to enable researchers to explore countermeasures against fabrication-time attacks. To the best of our knowledge, the existing targeted defensive IC layout techniques [13], [15], [16] are entirely *placement-centric*, i.e., filling unused space on the device layer with functional logic cells. While ICAS is capable of evaluating placement-centric defensive layout techniques, its security-insights also asses *routing-centric* defensive layout techniques. For example, layout engineers can leverage ICAS to create high degrees of routing congestivity in close proximity to security-critical nets. ICAS' security metrics enable IC layout designers to optimize the security of both the placement *and* routing of their designs.

*2) **Constrained Security Metrics**:* In its primary state, ICAS focuses on computing metrics that reason about the *spatial* resources required to implant hardware Trojans in IC layouts. While our metrics are unconstrained and thus conservative, it is trivial to extend, and constrain, ICAS metrics to account for other layout resources that may impact an attacker's decision process. For example, even with a plethora of spatial resources available to insert Trojan components, doing so in certain areas of the chip may impact local power consumption enough to disrupt normal operating behavior. Alternatively, inserting a hardware Trojan nearby un-shielded, fast toggling, interconnects may negatively impact the Trojan's signal integrity, rendering it benign. We recognize it is impractical to consider all possible constraints, and hence we design ICAS to be extensible.

*3) **Extensibility of Security Metrics**:* GDSII-Score is the ICAS tool that computes security metrics from an IC layout. It loads several files describing the IC layout to instantiate a single Python class (called "Layout") that contains queryable data structures containing a polygon representation of all components in the layout. Additionally, GDSII-Score contains several subroutines that compute spatial relationships between polygon objects and points within the layout. From these data structures and the provided subroutines, it is trivial to integrate additional metrics into GDSII-Score. To facilitate additional metrics, we open-source GDSII-Score [20], and our three example metrics that demonstrate how to query the main "Layout" data structure.

*4) **Extensibility of CAD Tools**:* Almost all steps of the IC design process utilize CAD tools. ICAS integrates into a commercial IC design process after placement-and-routing (Figure 1). While ICAS is validated with IC layouts generated by Cadence tools, integrating ICAS with other vendors' CAD tools does not require any additional effort due to the common process technology (LEF) and GDSII specifications used by ICAS.

*5) **Extensibility of Process Technologies**:* We test ICAS using IC layouts built with a $45\,nm$ SOI process technology; however, ICAS is agnostic of process technology. The LEF and layer map files (§VI) are the only ICAS input files that are dependent on the process technology. A LEF file describes the geometries and characteristics of each standard cell in the cell library, and the layer map file describes the layer name-to-number mappings, respectively, for a given process technology. ICAS adapts to different process technologies provided that all input files adhere to their specifications [37], [40].

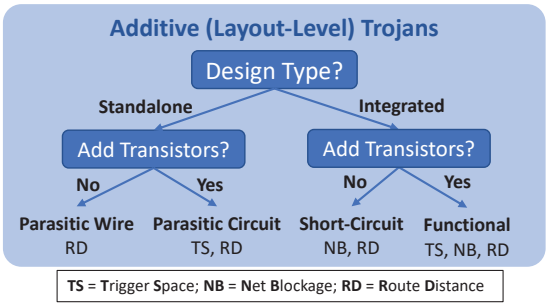

Fig. 12. We assume that, at the very least, layout-level additive Trojans require adding rogue wires to the layout[3]. Whether the Trojan design is *integrated* (requires connecting to a host circuit) or *standalone*, or requires *additional transistors*, the difficulty of inserting it into a victim IC layout can be captured by our three metrics: 1) Trigger Space (TS), 2) Net Blockage (NB), and 3) Route Distance (RD).

*6) **Limitations**:* The goal of ICAS is to estimate the susceptibility of circuit layouts to *additive* hardware Trojans, thus there are limitations. First, as implemented, ICAS is not capable of estimating the susceptibility of a circuit layout to *subtractive* or *substitution* Trojans. We are unaware of any stealthy and controllable subtractive hardware Trojans, but should researchers develop such an attack, metrics will need to be added to ICAS to enable detection. Dopant-level Trojans are the closest example of substitution Trojans [4], [5]. Though their non-existent footprints make them difficult to detect via side channels, post-fabrication imaging techniques that can identify such Trojans have been proposed [48]. Lastly, our implemented metrics do not capture the threat of via-only additive Trojans. A via-only attack shorts two vertically-adjacent wires for the purpose of leaking information. We feel the possibility of such pernicious attacks in the future highlights the importance of ICAS's extensibility.

*7) **Justification for Metrics**:* As a first step in estimating risk, we chose to implement three metrics that capture our decade worth of experience in implementing hardware Trojans: *net blockage*, *trigger space*, and *route distance*. These metrics capture the challenges we faced when inserting various types of additive Trojans into circuit layouts, i.e., *Trojan logic placement*, *victim/Trojan integration*, *intra-Trojan routing*. To facilitate mapping our metrics to specific Trojans we provide a taxonomy in Figure 12. To summarize the taxonomy: if a Trojan needs to attach to a victim wire (i.e., an integrated Trojan), our Net Blockage metric provides coverage; if the Trojan requires transistors to implement logic, our Trigger

[3]Via-only attacks are outside the scope of our metrics as they are currently implemented (§VIII-6).

Space metric provides coverage; and if the Trojan needs to be near the victim wire (for capacitive coupling in the case of a standalone Trojan or to meet timing requirements in the case of a integrated Trojan), our Route Distance metric provides coverage. Additionally, as our evaluation with existing Trojans and real IC layouts shows, our metrics are both Trojan- and IC-layout- sensitive. Lastly, the metrics are hardware design agnostic. While we do not suggest that the implemented metrics are all-encompassing, our results suggest that these metrics are a viable first step towards estimating a circuit's susceptibility to additive hardware Trojans.

IX. Related Work

Fabrication-time attacks and defenses have been extensively researched. Attacks have ranged in both size and triggering-complexity [4]–[6], [28], [29]. Defenses against these attacks include: side-channel analysis [9], [10], [12], [49], imaging [50], [51], on-chip sensors [52], [53], and preventive measures [13]–[16]. The most pertinent attacks and defenses are highlighted below.

A. Untrusted-foundry Attacks

The first foundry-level attack was conceived by Lin *et al.* [28]. This hardware Trojan was comprised of approximately 100 additional logic gates and designed to covertly leak the keys of an AES cryptographic accelerator using spread spectrum communication to modulate information over a power side channel. While the authors only demonstrated this attack on an FPGA, they are the first to mention the possibility of this type of Trojan circuit being implanted at an untrusted foundry.

The A2 attack [6] is the most recent fabrication-time attack. A2's analog triggering mechanism is stealthy, controllable, *and* small. It prevents the Trojan from being exposed during post-fabrication testing, or unintentionally through common usage. The attack requires only two additional standard cells and evades every known detection mechanism to date. ICAS quantifies the defensive coverage to these and other fabrication-time attacks.

B. Untrusted-foundry Defenses

Most untrusted foundry defenses rely on post-fabrication *detection* schemes [9], [10], [12], [49]–[53]. ICAS aims to guide innovation in *preventive* defenses against fabrication-time attacks, for which few mechanisms currently exist [13]–[16]. We highlight some of these preventive measures and how ICAS could measure their effectiveness.

While preventive security-by-design was first explored at the behavioral (RTL) level by of Jin *et al.* [42], Xiao *et al.* were the first to demonstrate security-by-design at the layout-level with their BISA (Built-In Self-Authentication) scheme [15]. The *undirected* BISA approach attempts to eliminate *all* unused space on the device layer placement grid, and create routing congestion, by filling the device layer with interconnected tamper-resistant fill cells. Alternatively, recognizing the impracticality of filling 100% of the empty placement sites in complex circuit designs, Ba *et al.* take a *directed* approach to filling empty placement cites [13], [16]. Specifically, they only fill empty placement sites in close proximity to security-critical nets.

X. Conclusion

ICAS is an extensible framework that we use to expose and quantify gaps in existing defenses to the threat posed by an untrusted foundry. ICAS has two high-level components: *Nemo*, a tool that bridges the semantic gap across IC design processes by tracking security-critical signals across all stages of hardware development and *GDSII-Score*, a tool that estimates the difficulty a foundry-level attacker faces in attacking security-critical logic. Experiments with over 60 IC layouts across three open-source hardware cores and four foundry-level hardware Trojans reveal that all current defenses leave the IC design vulnerable to attack—and some are totally ineffective. These results show the value of a tool like ICAS that can help designers identify and address defensive gaps.

From a high level, ICAS is momentus in that it makes security a first-class concern during IC layout (in addition to power, area, and performance): ICAS allows IC designers to measure the security implications of tool settings and design decisions. ICAS fits well with existing IC design tools and flows, allowing them to consider security. ICAS is a critical measurement tool that enables the systematic development of future physical-level defenses against the threat of an untrusted foundry.

Acknowledgment

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

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.

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported by the Under Secretary of Defense for Research and Engineering 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 Under Secretary of Defense for Research and Engineering.

References

[1] Intel Corporation, "Microprocessor quick reference guide," https://www.intel.com/pressroom/kits/quickreffam.htm.

[2] P. Alcorn, "Ice lake might arrive in june, according to leaked lenovo documents," https://www.tomshardware.com/news/lenovo-laptop-intel-ice-lake-10nm,38674.html.

[3] M. Lapedus, "Big trouble at 3nm," June 2018, https://semiengineering.com/big-trouble-at-3nm/.

[4] G. T. Becker, F. Regazzoni, C. Paar, and W. P. Burleson, "Stealthy dopant-level hardware trojans," in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2013.

[5] R. Kumar, P. Jovanovic, W. Burleson, and I. Polian, "Parametric trojans for fault-injection attacks on cryptographic hardware," in *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, 2014.

[6] K. Yang, M. Hicks, Q. Dong, T. Austin, and D. Sylvester, "A2: Analog malicious hardware," in *IEEE Symposium on Security and Privacy (SP)*, 2016.

[7] S. T. King, J. Tucek, A. Cozzie, C. Grier, W. Jiang, and Y. Zhou, "Designing and implementing malicious hardware," in *Usenix Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2008.

[8] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design & Test of Computers*, 2010.

[9] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using IC fingerprinting," in *IEEE Symposium on Security and Privacy (SP)*, 2007.

[10] Y. Jin and Y. Makris, "Hardware trojan detection using path delay fingerprint," in *IEEE Workshop on Hardware-Oriented Security and Trust (HOST)*, 2008.

[11] M. Potkonjak, A. Nahapetian, M. Nelson, and T. Massey, "Hardware trojan horse detection using gate-level characterization," in *ACM/IEEE Design Automation Conference (DAC)*, 2009.

[12] S. Narasimhan, X. Wang, D. Du, R. S. Chakraborty, and S. Bhunia, "TeSR: A robust temporal self-referencing approach for hardware trojan detection," in *IEEE Symposium on Hardware-Oriented Security and Trust (HOST)*, 2011.

[13] P.-S. Ba, S. Dupuis, M. Palanichamy, G. Di Natale, B. Rouzeyre *et al.*, "Hardware trust through layout filling: a hardware trojan prevention technique," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2016.

[14] R. P. Cocchi, J. P. Baukus, L. W. Chow, and B. J. Wang, "Circuit camouflage integration for hardware IP protection," in *ACM Design Automation Conference (DAC)*, 2014.

[15] K. Xiao and M. Tehranipoor, "BISA: Built-in self-authentication for preventing hardware trojan insertion," in *IEEE Symposium on Hardware-Oriented Security and Trust (HOST)*, 2013.

[16] P.-S. Ba, M. Palanichamy, S. Dupuis, M.-L. Flottes, G. Di Natale, and B. Rouzeyre, "Hardware trojan prevention using layout-level design approach," in *European Conference on Circuit Theory and Design (ECCTD)*, 2015.

[17] M. Hicks, M. Finnicum, S. T. King, M. M. K. Martin, and J. M. Smith, "Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically," in *IEEE Symposium on Security and Privacy (SP)*, 2010.

[18] H. Salmani, M. Tehranipoor, and R. Karri, "On design vulnerability analysis and trust benchmarks development," in *IEEE International Conference on Computer Design (ICCD)*, 2013.

[19] MIT Lincoln Laboratory, "Nemo," https://github.com/mit-ll/nemo.

[20] ——, "GDS2-Score," https://github.com/mit-ll/gds2-score.

[21] A. Waksman, M. Suozzo, and S. Sethumadhavan, "FANCI: identification of stealthy malicious logic using boolean functional analysis," in *ACM SIGSAC Conference on Computer & Communications Security (CCS)*, 2013.

[22] H. Salmani and M. Tehranipoor, "Analyzing circuit vulnerability to hardware trojan insertion at the behavioral level," in *IEEE Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2013.

[23] R. S. Chakraborty, F. G. Wolff, S. Paul, C. A. Papachristou, and S. Bhunia, "MERO: A statistical approach for hardware trojan detection." in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2009.

[24] M. Rostami, F. Koushanfar, J. Rajendran, and R. Karri, "Hardware security: Threat models and metrics," in *IEEE International Conference on Computer-Aided Design (ICCD)*, 2013.

[25] M. Beaumont, B. Hopkins, and T. Newby, "Hardware trojans-prevention, detection, countermeasures (a literature review)," Defence Science and Technology Organization Edinburgh (Australia), Tech. Rep., 2011.

[26] R. S. Chakraborty, S. Narasimhan, and S. Bhunia, "Hardware trojan: Threats and emerging solutions," in *IEEE High Level Design Validation and Test Workshop (HLDVT)*, 2009.

[27] F. Wolff, C. Papachristou, S. Bhunia, and R. S. Chakraborty, "Towards trojan-free trusted ICs: Problem analysis and detection scheme," in *ACM Conference on Design, Automation and Test in Europe (DATE)*, 2008.

[28] L. Lin, M. Kasper, T. Güneysu, C. Paar, and W. Burleson, "Trojan side-channels: Lightweight hardware trojans through side-channel engineering." in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2009.

[29] Y. Shiyanovskii, F. Wolff, A. Rajendran, C. Papachristou, D. Weyer, and W. Clay, "Process reliability based trojans through NBTI and HCI effects," in *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, 2010.

[30] M. Hicks, C. Sturton, S. T. King, and J. M. Smith, "SPECS: A lightweight runtime mechanism for protecting software from security-critical processor bugs," in *ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2015.

[31] X. Wang, M. Tehranipoor, and J. Plusquellic, "Detecting malicious inclusions in secure hardware: Challenges and solutions," in *IEEE Workshop on Hardware-Oriented Security and Trust (HOST)*, 2008.

[32] T. Force, "High performance microchip supply," *Defense Technical Information Center (DTIC), Annual Report*, 2005.

[33] H. Salmani, "COTD: Reference-free hardware trojan detection and recovery based on controllability and observability in gate-level netlist," *IEEE Transactions on Information Forensics and Security*, 2017.

[34] J. Zhang, F. Yuan, L. Wei, Y. Liu, and Q. Xu, "VeriTrust: Verification for hardware trust," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2015.

[35] L. H. Goldstein and E. L. Thigpen, "SCOAP: Sandia controllability/observability analysis program," in *ACM Design Automation Conference (DAC)*, 1980.

[36] OpenCores.org, "OpenRISC OR1200 processor," https://github.com/openrisc/or1200.

[37] Cadence Design Systems, *LEF/DEF Language Reference*, 2009, http://www.ispd.cc/contests/14/web/doc/lefdefref.pdf.

[38] E. Sperling, "Design rule complexity rising," April 2018, https://semiengineering.com/design-rule-complexity-rising/.

[39] W. C. Elmore, "The transient response of damped linear networks with particular regard to wideband amplifiers," *Journal of Applied Physics*, 1948.

[40] Cadence Design Systems, *Layer Map Files*, http://www-bsac.eecs.berkeley.edu/ cadence/tools/layermap.html.

[41] Calma Company, *GDSII Stream Format Manual*, February 1987.

[42] Y. Jin, N. Kupp, and Y. Makris, "DFTT: Design for trojan test," in *IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, 2010.

[43] T. Linscott, P. Ehrett, V. Bertacco, and T. Austin, "SWAN: mitigating hardware trojans with design ambiguity," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2018.

[44] R. Zhang, N. Stanley, C. Griggs, A. Chi, and C. Sturton, "Identifying security critical properties for the dynamic verification of a processor," in *ACM Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2017.

[45] S. Williams, "Icarus Verilog," http://iverilog.icarus.com/.

[46] J. F. Hughes and J. D. Foley, *Computer graphics: principles and practice*. Pearson Education, 2014.

[47] MIT Lincoln Laboratory, "Common evaluation platform," https://github.com/mit-ll/CEP.

[48] T. Sugawara, D. Suzuki, R. Fujii, S. Tawa, R. Hori, M. Shiozaki, and T. Fujino, "Reversing stealthy dopant-level circuits," in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2014.

[49] J. Balasch, B. Gierlichs, and I. Verbauwhede, "Electromagnetic circuit fingerprints for hardware Trojan detection," in *IEEE International Symposium on Electromagnetic Compatibility (EMC)*, 2015.

[50] B. Zhou, R. Adato, M. Zangeneh, T. Yang, A. Uyar, B. Goldberg, S. Unlu, and A. Joshi, "Detecting hardware trojans using backside optical imaging of embedded watermarks," in *ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2015.

[51] R. Adato, A. Uyar, M. Zangeneh, B. Zhou, A. Joshi, B. Goldberg, and M. S. Unlu, "Rapid mapping of digital integrated circuit logic gates via multi-spectral backside imaging," *arXiv:1605.09306*, 2016.

[52] J. Li and J. Lach, "At-speed delay characterization for IC authentication and trojan horse detection," in *IEEE Workshop on Hardware-Oriented Security and Trust (HOST)*, 2008.

[53] D. Forte, C. Bao, and A. Srivastava, "Temperature tracking: An innovative run-time approach for hardware trojan detection," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2013.

APPENDIX A
ROUTE DISTANCES OF OR1200 LAYOUTS

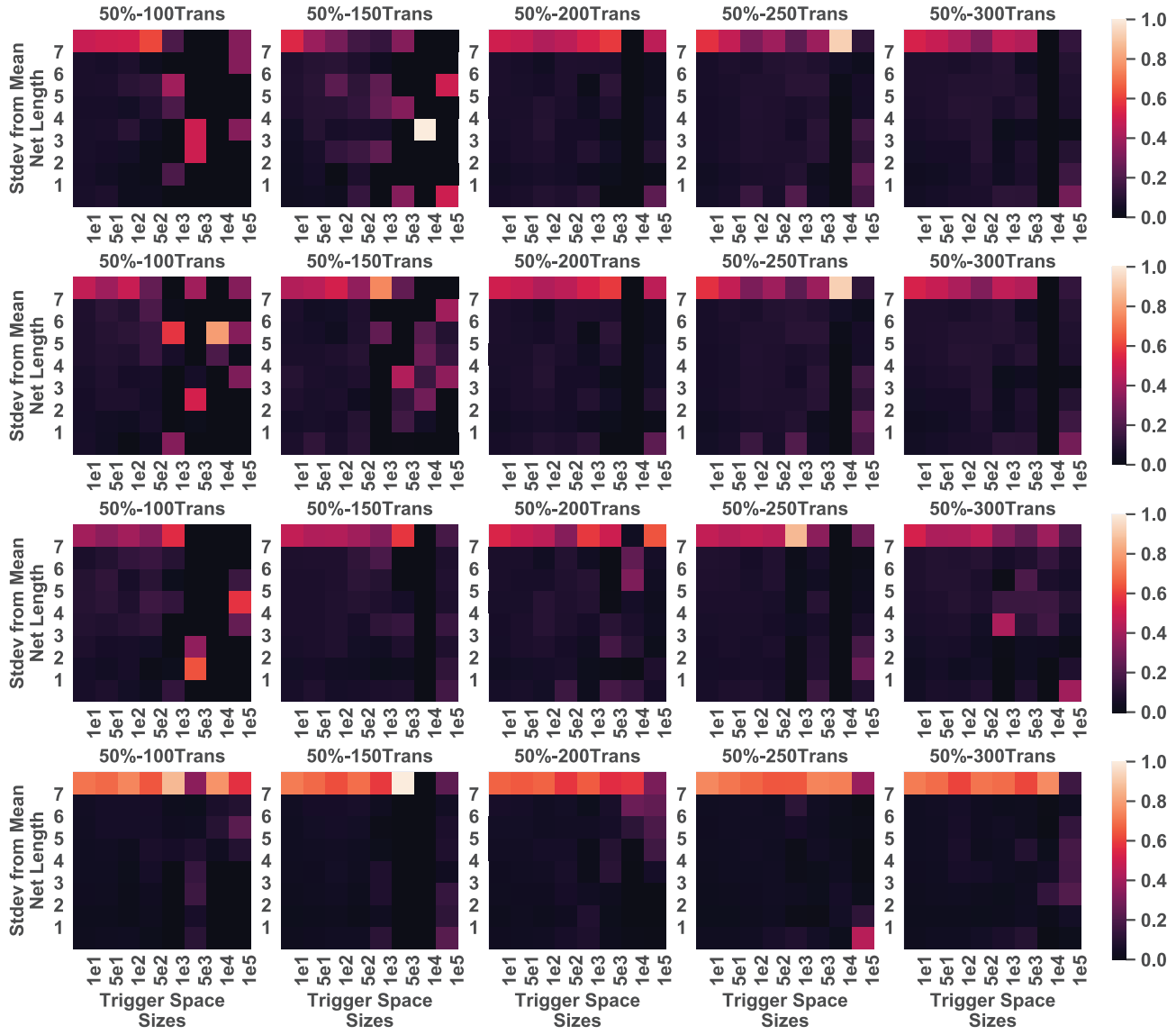

Fig. 13. *Route Distance Metric for OR1200 at 50% Density).* A target density of 50% was held across each layout, while target clock frequency and max transition time parameters were varied from 100 MHz to 1000 MHz and 100 ps to 300 ps respectively. Each heatmap is intended to be read column-wise, where each column is a histogram. The color intensity within a heatmap column indicates the percentage of (critical-net, trigger-space) pairs, within that column, that are within a range of distance away. The y-axis reports the distance in terms of standard deviations from the overall mean net-length in each design. The x-axis reports the trigger space sizes in number of contiguous placement sites. Designs with smaller trigger-spaces and long route distances are more resistant to fabrication-time attacks. Namely, a heatmap column that is completely dark indicates no (critical-net, trigger-space) pairs, or attack points, and a column that is completely dark except for the top-most cell is the second most secure.

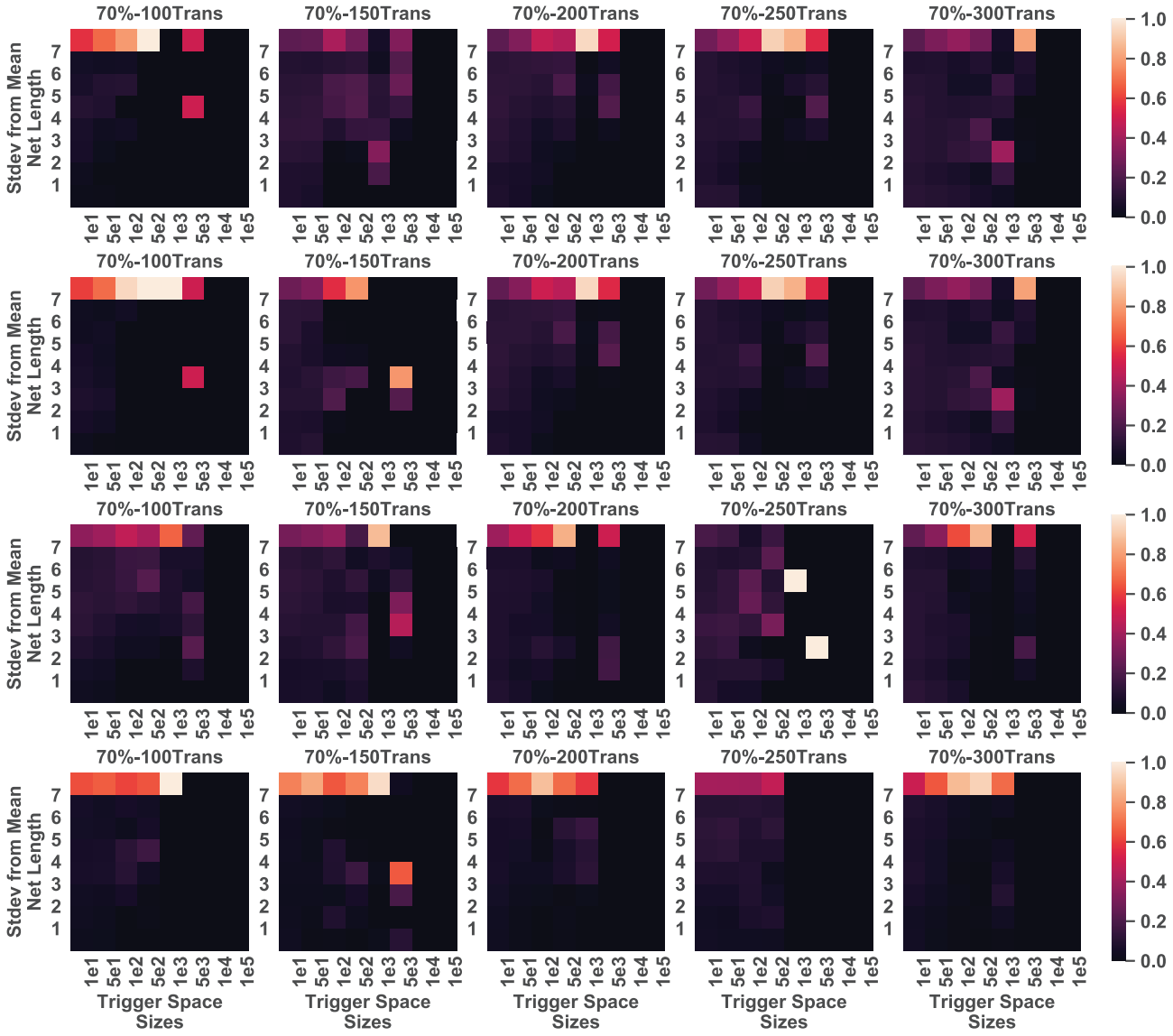

Fig. 14. *Route Distance Metric for OR1200 at 70% Density.* Same as Fig. 13, except a target density of 70% was held across each layout.

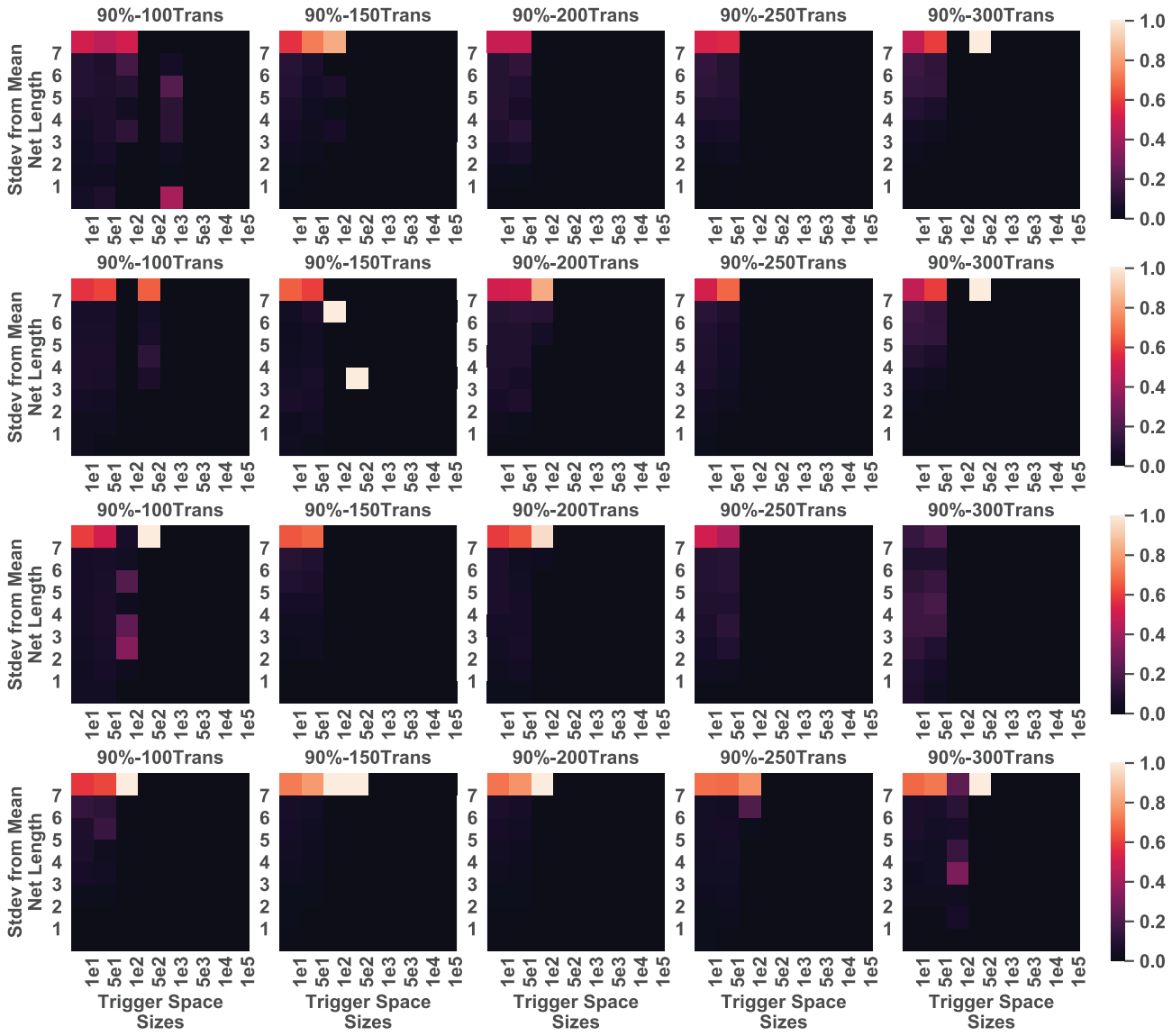

Fig. 15. *Route Distance Metric for OR1200 at 90% Density.* Same as Fig. 13, except a target density of 90% was held across each layout.