DEFending Integrated Circuit Layouts

JITENDRA BHANDARI* and JAYANTH GOPINATH*, New York University, USA
MOHAMMED ASHRAF, NYU Abu Dhabi, UAE
JOHANN KNECHTEL, NYU Abu Dhabi, UAE
RAMESH KARRI, New York University, USA

The production of modern integrated circuit (IC) requires a complex, outsourced supply chain involving computer-aided design (CAD) tools, expert knowledge, and advanced foundries. This complexity has led to various security threats, such as Trojans inserted by adversaries during outsourcing, and physical probing or manipulation of devices at run-time. Our proposed solution, *DEFense* is an extensible CAD framework for evaluating and proactively mitigating threats to IC at the design-time stage. Our goal with *DEFense* is to achieve "security closure" at the physical layout level of IC design, prioritizing security alongside traditional power, performance, and area (PPA) objectives. *DEFense* uses an iterative approach to assess and mitigate vulnerabilities in the IC layout, automating vulnerability assessments and identifying vulnerable active devices and wires. Using the quantified findings, *DEFense* guides CAD tools to re-arrange placement and routing and use other heuristic means to "DEFend" the layouts. *DEFense* is independent of back-end CAD tools as it works with the standard DEF format for physical layouts. It is a flexible and extensible scripting framework without need for modifications to commercial CAD code bases. We are providing the framework to the community and have conducted a thorough experimental investigation into different threats and adversaries at various stages of the IC life-cycle, including Trojan insertion by an untrusted foundry, probing by an untrusted end-user, and intentionally introduced crosstalk by an untrusted foundry.

Additional Key Words and Phrases: Hardware Security, Integrated Circuits, Physical Design, Trojans, Probing Attacks, Defense, Security Assessment

ACM Reference Format:
Jitendra Bhandari, Jayanth Gopinath, Mohammed Ashraf, Johann Knechtel, and Ramesh Karri. 2023. DEFending Integrated Circuit Layouts. In . ACM, New York, NY, USA, 23 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Due to the complexity of developing advanced IC, a distributed supply chain model has emerged. To avoid the high costs of building and maintaining a semiconductor foundry for advanced nodes, many design companies now use the fabless design model. Under this approach, the final design layout is sent overseas in GDSII format for fabrication.

Unfortunately, the design team's lack of control over external facilities as shown in Fig. 1 leaves the IC open to malicious modifications, such as so-called Trojans inserted into the final design layout. This may cause the IC to malfunction at mission-critical operations, leak confidential information, or reduce its overall reliability [16]. Furthermore, secure data like cryptographic keys can be extracted from the design's nets or instances through methods like Focused Ion Beams (FIB) or similar techniques. These attacks are feasible even after the fabrication and testing of the IC, as shown in Fig. 1. To extract secret information, an attacker can use a functional IC and perform probing-based attacks.

*Both authors contributed equally to this research.

Manuscript submitted to ACM

Fig. 1. Design flow and threats. Green stages are secure and Red stages are vulnerable to attacks.

Additionally, [10] proposed a vulnerability that exploits capacitive coupling's parasitic phenomenon to trigger a signal on the wire of interest, causing the leak of sensitive keys or the exploitation of privilege escalation.

Given the size and complexity of layouts, conducting a comprehensive assessment against these and other threats becomes a practical challenge. Even if vulnerabilities are discovered, addressing them by sending the results to the team for layout modification is challenging, considering that the modern IC supply chain has fast-paced development and time-to-market constraints. Therefore, an in-house solution may be necessary to tackle these issues.

In industry, engineers employ custom scripts to fix issues that arise in the layout. This allows them to analyze and interact with CAD tools to add fixes. We propose to develop a similar framework with integrated evaluation tools and DEFense strategies into the physical design flow, as shown in Fig. 2, based on the Design Exchange Format (DEF) and other files. The DEF format is an intermediate representation of all the physical parameters of the layout, cells, and routes in the layout. The files along with the design netlist can be used to load designs into any of the leading commercial CAD tools. We write evaluation and defense scripts that can be ported between physical design tools like Innovus from Cadence and ICC2 from Synopsys. DEF allows easy visualization of vulnerabilities to the designer so that they can target their fixes to specific areas to secure the layouts against different attacks with little logic overhead. We demonstrate the evaluation and defense scripts using the Cadence tools on MIT Lincoln Lab Common Evaluation Platform (CEP) system-on-chip (SoC) Chip and OR1200 processor layouts. We scan the vulnerabilities in the baseline implementations produced using standard CAD flows. Our approach encourages designers to prioritize layout security during the design process [11]. Our contributions are threefold:

(1) Early evaluation of empty trigger space in Trojan-based scenarios during the Placement Stage, along with defense algorithms that have little impact on PPA.
(2) Evaluation of the exposed area of nets and instances to probing-based attacks and implementation of defense algorithms that ensure layout security without affecting PPA targets.
(3) Evaluation of the layout using proposed defenses against crosstalk-based attacks.

Section 2 discusses previous efforts to address the problem, while Section 3 provides a Physical Design flow overview and explains our work's rationale. Our *DEFense* methods are detailed in Section 4, and Section 5 reports the results of our experimental setup and analysis. Finally, we present our conclusions and insights from the study in Section 6.

2 RELATED WORK

There has been limited research on attacks and defenses at the layout level of IC. Some studies have focused on developing Trojans that are difficult to detect, while few have addressed defenses [8]. [1, 2] proposed defense method involves filling empty spaces in the layout at the end of the implementation process to facilitate the detection of Trojans inserted by untrusted foundries. Another approach utilizes a ring oscillator network to detect modifications in the layout at the untrusted foundry by measuring the change in the delay of the oscillator circuit [20]. Delay fingerprinting can also be applied to specific paths in the layout to identify changes in different data paths. However, these defenses are not entirely reliable due to variations in On-Chip and process-based delays. Another approach for addressing the

issue involves implementing Built-in self-authentication (BISA), which replaces filler cells with test logic to ensure the layout hasn't been tampered with during post-silicon testing, as described in [17]. However, this solution is vulnerable to attack if a malicious actor modifies the connections to the output response analyzer circuit to obtain the correct output and modify the empty spaces without detection. Another solution, proposed in [2], involves inserting shift registers after Place-And-Route (P&R) to verify special functions in empty placement sites. While these cells are inactive during functional mode and are used only during post-silicon testing, they may have side effects on the PPA due to additional dynamic power on the clock paths to shift registers.

For front-side probing attacks, modifications made to the layout are considered to defend against different types of probing at various angles. However, there are no existing tools that can determine the exposure level of security-critical nets and instances to front-side probing-based attacks. Some studies have developed customized techniques to assess net exposure and recommend solutions such as active shields and t-private circuits. A physical design flow was suggested in [15], which introduced additional steps to guide floorplanning and routing, but this approach has overhead in terms of additional logic and power, and does not consider probing the base layer of the instance to obtain its value. Modifying the layout is often done after the physical design flow, making it challenging to add and route extra logic without interfering with functional paths. [13] has suggested a range of defenses against such attacks, including special packaging and active shielding, but there are limited optimization possibilities post-implementation, making it challenging to balance increased layout security with better PPA metrics. For complex layouts, making changes to the design may require manual adjustments that could impact tight deadlines and introduce unpredictability.

There are limited tools available to assess the security of IC layouts. One study, as described in [14], scans a GDSII file and identifies security-critical nets specified by the user to generate a score indicating the vulnerability of the layout to Trojans insertion, probing, and distance between the Trojan cells and the security-critical nets, while adhering to DRC guidelines and timing checks. Although this tool shows promise, the results it provides cannot be easily integrated into CAD tools. The designer must examine the text reports and add the necessary fixes to the final layout, which can be challenging for complex designs without affecting other parts of the design.

DEFense integrates evaluation and defenses for improving layout security into the physical design flow, resulting in an automatic enhancement of security throughout the flow. Our results show that the defenses have negligible impact on PPA while significantly reducing the number of vulnerabilities compared to the baseline layouts.

3 BACKGROUND

3.1 IC Physical Design Flow

The IC physical design flow is utilized to create layouts in the GDSII format, detailing the various shapes of metal and silicon layers that are required to be etched through photo-lithographic masks. Logic gates are represented through standard cells on the base layer of the IC, with pin connections typically implemented using lower metal layers. Higher metal layers are utilized for routing connections between cells, while special cells such as hard intellectual property block (IP) and additional power and clock routes are implemented in an optimal fashion through the physical design flow. As shown by the blue-colored boxes in Fig. 2, the standard Physical Design flow is illustrated [16].

The primary design input for the IC physical design flow is the register transfer level (RTL) files, which are written in a hardware description language (HDL) like Verilog or VHDL. These files describe the behavior of the design for various input conditions and the necessary response for each internal state, enabling the module to achieve the required functionality. In a secure design environment, the entire design is verified to confirm that the design functions as

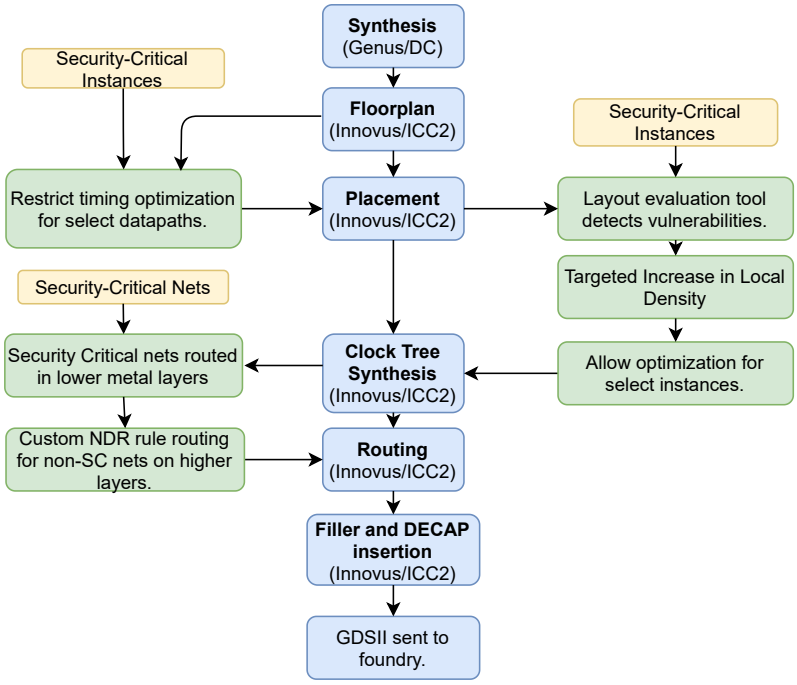


Fig. 2. Physical Design Flow with DEFense steps integrated to CAD flow.

expected, and there are no bugs in the RTL code. Third-party IP integrated into the RTL is also verified to ensure that they meet the design's functional requirements [7]. Ideally, the overall design should be verified to guarantee that no additional logic is introduced that could be exploited for malicious purposes. Various other works in the literature have explored the implementation and detection of Hardware Trojans at the RTL stage [4, 19].

After completing the design verification, the design is synthesized using electronic design automation (EDA) tools. This involves transforming RTL designs into a gate-level netlist, which is achieved by utilizing a standard cell library corresponding to a specific technology node. The gate-level representation of the design has identical functionality to the RTL design, which is verified through either Logical Equivalence Checks (LEC) or gate-level simulation with varied input vectors. The netlist furnishes information about the interconnections between each cell within the design, serving as the primary phase in the hardware implementation process. The EDA tool is leveraged to optimize the design for PPA of the hardware design by utilizing different targets and tool settings to obtain the optimal implementation.

Subsequently, the netlist is fed into another set of tools to facilitate the layout design process. The first step is to develop a floorplan, which entails determining the shape and dimensions of the IC and its sub-blocks. The IC size is dependent on the number of standard cells in the design and other hard macro components, with additional area overhead being necessary for routing the nets between different standard cells. Using the area size from the input netlist, a target area and size for the required design utilization are established. At this stage, the locations of various I/O interfaces are also fixed, and rows are created to accommodate the different components of the design. Next, the hard macros in the design, such as memory elements, sub-blocks, and IP are placed within the design. Finally, the power grid is constructed for both the rows of standard cells and the hard macros in the block.

Next, the standard cells are positioned within these rows, which should be optimized to fulfill all the design constraints with minimal area and power consumption. Once the initial placement is complete, the nets are optimized to comply with signal transition time, load, and fanout requirements. An estimation is performed to verify the efficient routing of the design without encountering any issues. If any areas exhibit routing congestion, the tools adjust the placement to minimize the cell and pin density in that area. The cells are subsequently sized to satisfy the setup timing requirement. This iterative process continues until the design meets the required objectives.

Once the placement of all the cells is done, the clock tree is constructed to provide a low-skew clock signal to all sequential elements in the design. It is important to maintain clock transitions to prevent the clock signal quality from degrading later during the signal routing step. This is achieved by employing special clock cells with balanced rise and fall transitions, and using wider nets. Any violations of setup and hold timing that may arise at this point are subsequently resolved [3].

In the routing step, all signals in the design are routed to satisfy the design rule requirements (DRCs) with little detours to achieve the best possible performance and meet the required PPA targets. Additionally, tools are used to guarantee signal integrity by minimizing the effects of any signal changes on neighboring nets [3]. Once all the necessary design goals have been accomplished, any remaining empty spaces in the layout are filled with filler and DECAP cells. These cells have no specific functionality but are necessary to meet fabrication requirements for n-well continuity. After the completion of these steps, the final GDSII file is created and forwarded to the foundry for fabrication.

3.2 Use of White Spaces in IC layouts

Table 1. Increase in utilization of a simple AES module through the Physical Design FLow

Physical Design Flow Stage	Utilization
Floorplan	70%
Placement	74.47%
Clock Tree Synthesis	75.12%
Routing	75.8%

Large-scale commercial designs often have substantial amounts of unoccupied space, providing opportunities for malicious parties to insert Trojans. Typically, IC layouts require extra space for routing between standard cells to ensure that each net of the design can be routed while adhering to the DRC requirements of a given technology node. This requires additional area overhead to the baseline standard cell area. In determining the final size of an IC, there are other factors to consider besides routing overhead area. The amount of additional space required compared to the synthesized standard cell area varies at each stage of the physical design process. Table 1 illustrates the increase in design density across different stages of the design flow.

In the floorplanning stage, physical-only ENDCAP and TAP cells are inserted to prevent latch-up and ensure proper integration of the die in compliance with base layer design rules. In case the design requires power gating, power switch cells from commercial libraries are incorporated into the layout. This results in the need for extra white space overhead beyond the baseline synthesized standard cell area.

During the placement stage, the high fanout signal trees are reconstructed based on the actual placement of cells. A high fanout signal may need to drive thousands to millions of cells, depending on the size of the IC. Therefore, it is crucial to have an effective tree-like structure to ensure that all the cells in the fan-out of this net receive the input

signal with low transition time, good signal integrity, and meet timing requirements. To achieve this, buffer insertion and cell upsizing are performed to meet signal transition time, fanout, and load requirements.

Next, a balanced clock tree is constructed to guarantee low clock skew across the various corners of the chip for all sequential elements. Since modern IC has millions of sequential elements controlled by the clock signal, a considerable number of inverters/buffers are added to decrease skew and satisfy clock transition requirements. Skew problems may result in hold time violations in the design, which can be resolved by adding buffers to short data paths. This, once again, necessitates white spaces near these paths.

After resolving any hold time violations, the signal routing process begins. High-priority signals like clock and timing-critical signals are routed first to minimize detours and delays, with the remaining signal nets being routed afterward. Since high-priority nets block routing tracks, the lower-priority nets may need to take detours from their optimal path, increasing the load on the nets and degrading signal transition times. To prevent timing violations elsewhere, buffers are added to address any violations resulting from the detours. Additional white space is necessary to complete each physical design stage optimally. However, any remaining white spaces at the end of the physical design flow are filled with non-functional filler cells.

3.3 Hardware Trojans in IC Layouts

After the final layout is sent to the foundry, it is possible to add Trojans at the layout level by substituting non-functional filler cells with standard or analog logic cells to form the trigger for the Trojan. It is difficult to detect the replacement of these filler cells, especially if the Trojan is designed to be stealthy in size and trigger condition. Trojans can target security-critical design nets/registers by routing them to the Trojan circuit. The number of placement sites required for Trojan insertion depends on the type of Trojan, with privilege escalation and key extraction Trojans requiring 342 and 2552 placement sites [9], respectively. A2 Analog Trojan [18] requires only two standard cells or 20 placement sites and can alter a signal's value by repeatedly activating the trigger input until it reaches a given threshold, which causes the circuit output to hold a constant logic '1', regardless of changes in the input.

To ensure that additive Trojans do not affect the existing design circuitry's timing requirements, it's important to place and route the additional logic appropriately. Standard cells have varying delays based on their output load and input signal transition time. When the baseline design is sent to the foundry, all paths must meet these timing requirements. If Trojans and routes for the payload deteriorate the signal transition time or increase the load on the standard cells, it may violate the timing requirements of the design, which would be detected during post-silicon testing when an incorrect output is captured at the end of the clock cycle. In order for an attack to be successful, it is crucial that the timing requirements of the baseline design are still met even after the Trojan has been inserted. When it comes to additive Trojans, standard cells or analog circuitry are necessary components of the design. However, as the size of these Trojans decreases, detecting them using side-channel analysis in post-silicon designs becomes increasingly difficult. While destructive layer-by-layer imaging and comparison with a golden IC layout can be used to detect these cells and routes, small and covert Trojans that are inserted into a small subset of fabricated ICs at the foundry level may go undetected, as noted in [12].

3.4 Probing of Security Critical Signals

Hardware implementations of cryptographic modules and security-critical accelerators have been shown to significantly improve performance and reduce overhead from the main processor. In these implementations, data confidentiality is achieved by encrypting sensitive data using a secret key, which can be stored in registers or tamper-resistant ROMs in

the layout during operation. However, this approach can leave the key vulnerable to probing attacks, as the nets used for routing the key or the registers storing the key value can be targeted using a focused ion beam (FIB) to extract the secure key value. It is assumed that the attacker has full layout information and knowledge of the precise location of the nets or instances to be targeted for the probing attack. It is important to note that we are only considering defenses and evaluation of front-side probing attacks. While existing countermeasures such as active shields and special packaging can help protect against such attacks, they can lead to additional overhead in terms of power, performance, and cost for advanced packaging methodologies. Even methods integrated into physical design flows can lead to some degradation in terms of PPA, as described in [15].

3.5 Crosstalk

From an IC design perspective, capacitive coupling is a significant challenge that arises due to the parasitic component of running metal wires in parallel, as noted in [3]. When voltage is toggled in two different nets in neighboring wires, it results in a sharing of charge. This effect can be modeled by a small coupling capacitor, as shown in Fig. 3. In advanced technology nodes, where the minimum distance between two wires is reduced, this issue becomes increasingly prominent. As a result, there is a significant coupling capacitance between nets that are routed in parallel. The expression for capacitance can be obtained as:

$$C = \epsilon \frac{A}{d}$$

where, ϵ is the permittivity of the medium, A is the effective area shared between two net routes and d is the minimum separation between them.

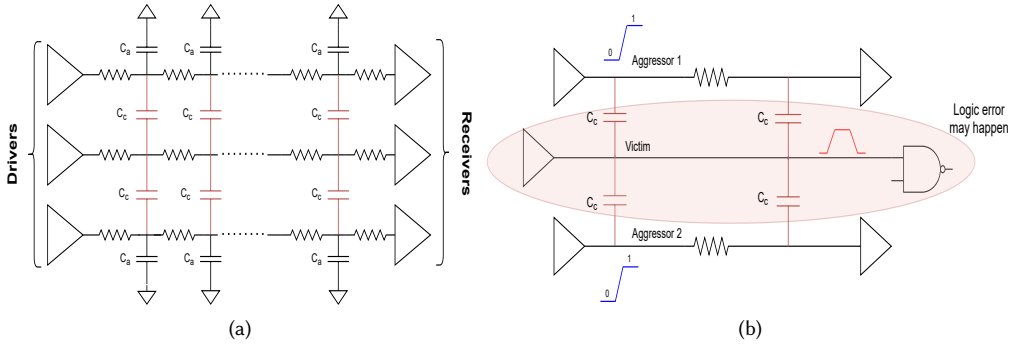


Fig. 3. (a) Cross-Coupling Capacitors arise due to running of metal wires in close proximity. C_a is the effective wire capacitance, and C_c is the coupling capacitance between metal wires. (b) Victim net which is affected due to toggling of signals in aggressor nets.

When a net has a large amount of coupling capacitance, changing the signal from one logic state to another can affect the arrival time of the signal in another net. This phenomenon is called crosstalk, and it occurs when one metal route interferes with another. These changes in the delay values can cause setup or hold timing violations on the datapath. Additionally, the capacitive coupling effect could also induce a small voltage change in another net that is routed in close proximity to the original net. If the magnitude of this change in voltage is significant enough to cause the signal to be captured at the endpoint with its logic state flipped, the design functionality might be affected, as discussed in [3].

A new threat model presented in [10] leverages crosstalk to trigger a signal for a privilege escalation attack in an System-on-Chip (SoC). This type of attack poses a new threat without the need for additional gates to trigger the Trojan logic. This results in a stealthy and hard-to-detect Trojan.

4 DEFENSE METHODOLOGY

Scripting languages such as Tcl and perl can be easily integrated into existing CAD tools. These scripts are used widely in existing physical design flows. These scripts can be used to enable the designers to analyze the layouts, and to evaluate vulnerabilities in the layout to attacks relevant to the functionality of the design. Our evaluation and defense framework is scalable and tool-agnostic. It can be ported to work on different designs across tools and technology nodes. Leading CAD layouts tools like Cadence Innovus and Synopsys ICC2 support Tcl-based scripts. We wrote the design flows and custom scripts for layout DEFense using Tcl-based commands and scripts. By using these scripts we can evaluate the vulnerabilities in the layout and quantifies the magnitude of the threat in the layout. Due to the easy interface with the CAD tool, the scripts can automatically highlight the layout to show the vulnerable areas and the designers can apply defenses at any stage in the physical design flow. **Finally, since DEFense scripts are extensible, new layout-level attacks in the future can be easily modeled and evaluated.**

To demonstrate the ease of integration of the results from our scripts into the physical design flow, we develop a defense approach where the evaluation scripts can be run at different stages of the physical design flow. This can identify vulnerabilities in the layout near security critical logic or in the entire design as per the designer's guidance. In turn, the tool automatically applies defenses to reduce the number of vulnerable sites in the layout. Similarly, for probing attacks, it applies preemptive fixes during the physical design flow based on the input security critical nets provided by the user. The evaluation and defense scripts can easily be ported to different flows, designs, and tools to optimize layout-level security alongside PPA. Fig. 2 represents the standard Physical design flow with the modifications that we have made to obtain layouts with much lower vulnerability and with negligible PPA impact on the design.

4.1 Layout Level Trojan Insertion

4.1.1 Assessment. In security-critical cryptographic accelerators, a hardware Trojan-based attack does not have to be limited to a key extraction attack. Previous work has shown that a single-bit fault can break the AES implementation in hardware [6]. Similarly, for DSP accelerators, a single bit flip on the "next_out" signal which indicates completion of the computations can return incorrect results. This is true for the "supv" signal in the OR1200 processor as well. By changing the value of this signal privilege escalation attacks can be deployed. A stealthy A2 Analog Trojan which can cause a single-bit flip and cause incorrect results requires only 20 placement sites [18].

Due to our fully generic and parameter-based scripts, the minimum threshold for the number of placement sites to be considered for Trojan Insertion can be set as per the user requirement. We evaluate all the layouts with the most pessimistic case of 20 placement sites for the A2 Analog Trojan. These scripts can be utilized on any given layout for the user to visualize the vulnerabilities in the layout and analyze the generated reports or they can be integrated into the CAD flow for driving the layout defense optimization.

For a site to be considered vulnerable during the implementation phase in the CAD flow, we consider any unused placement site in the design as vulnerable. After the implementation is complete, we consider any filler cell placed in the design as vulnerable as it can be replaced by the attacker to add Trojan logic. The radius from the security-critical logic to be considered for evaluation can be set by the user for distance-based evaluation as shown in line 1 of Algorithm 1. Since the Trojan needs to be placed and routed such that the timing requirements of the original design are still met,

Algorithm 1: Distance-based Evaluation and Defense against Trojan Insertion

Input: Security-critical key registers SK, Radius R
Output: Report $file$

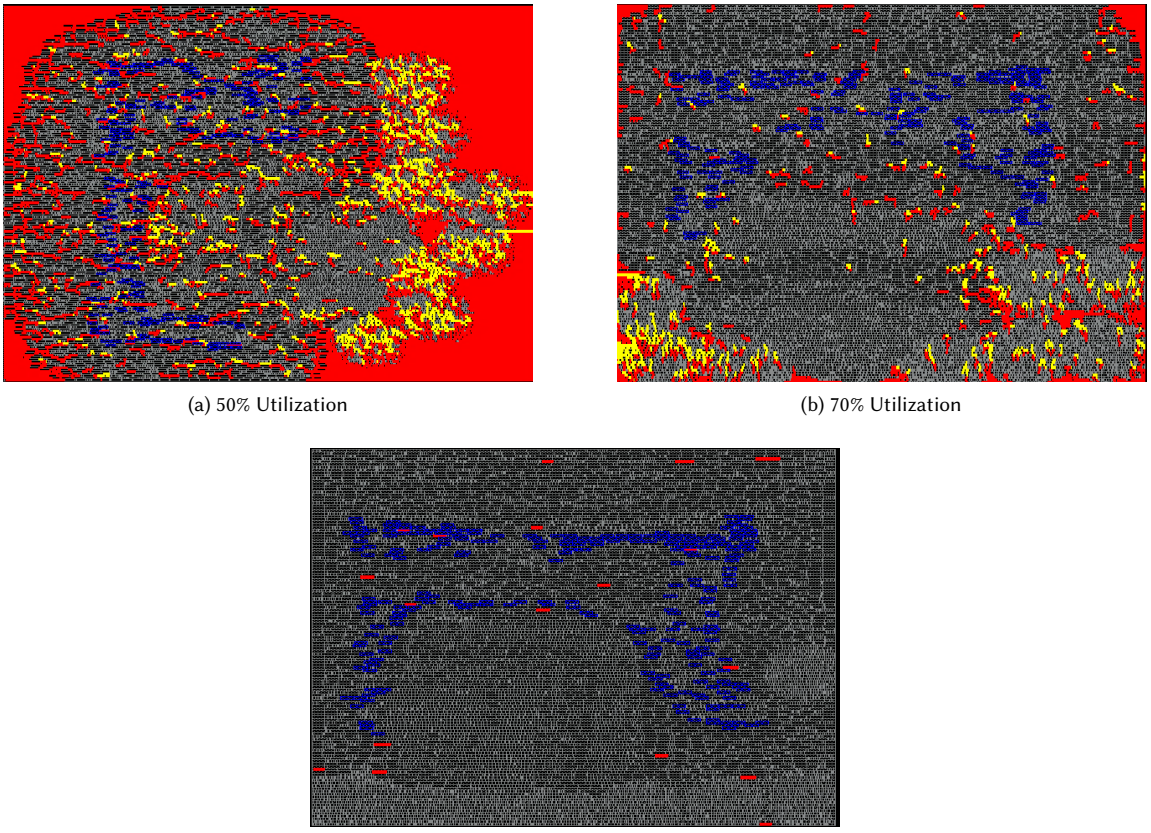
Function CalculateVulnerableRegion(s, R):
    $place\_site \leftarrow$ GetAvailablePlacementSites(s, r); // Free sites within radius r of s
    $sort\_dist \leftarrow$ Sort($place\_site$); // Placement site in increasing order of distance from s
    $sort\_route \leftarrow$ Sort($place\_site$); // Placement site in decreasing order of available routing resource
    $Regions \leftarrow sort\_dist \cap sort\_route$;
    return $Regions$; // Return Vulnerable Regions
$Before\_Defense \leftarrow \emptyset$;
foreach $s \in SK$ **do**
    $Before\_Defense[s] \leftarrow CalculateVulnerableRegion(s, R)$;
end
foreach $s \in SK$ **do**
    Create regions and bounds in the radius R of s ; // Increase local density
    Create partial blockages in the radius R of s ; // Even distribution of logic
    Run ; // Tool runs with mentioned changes
end
$After\_Defense \leftarrow \emptyset$;
foreach $s \in SK$ **do**
    $After\_Defense[s] \leftarrow CalculateVulnerableRegion(s, R)$;
end
$file \leftarrow \{Before\_Defense, After\_Defense\}$; // Return the vulnerable regions before and after defense
return $file$

Algorithm 2: Timing-based Evaluation and Defense against Trojans

Input: Security-critical paths SP
Output: Report $file$

$delay \leftarrow STDLIB(NAND)$; // Cost of adding a NAND in terms of routing and propagation delay
$Before\_Defense \leftarrow \emptyset$;
foreach $s \in SP$ **do**
    $len \leftarrow DISTANCE(s, delay)$; // longest distance possible with Trojan and timing met
    $Before\_Defense[s] \leftarrow CalculateVulnerableRegion(s, len)$; // From Algorithm 1
end
foreach $s \in SP$ **do**
    Break complex cells in s into simple cells; // Increase logic level
    Decrease drive strength of this cells;
    Set dont_touch attribute on the cells ; // Stop tool from optimizing this cells
end
$After\_Defense \leftarrow \emptyset$;
foreach $s \in SP$ **do**
    $len \leftarrow DISTANCE(s, delay)$;
    $After\_Defense[s] \leftarrow CalculateVulnerableRegion(s, len)$;
end
$file \leftarrow \{Before\_Defense, After\_Defense\}$; // Return the vulnerable regions before and after defense
return $file$

placement sites which are closer to security critical logic are more vulnerable to Trojans in comparison to placement sites that are much further away. Also, among the placement sites which are near the security critical logic, the sites over



(a) 50% Utilization

(b) 70% Utilization

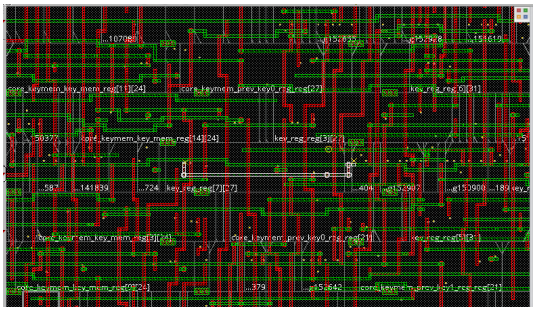
(c) 90% Utilization

Fig. 4. Visualization of AES layout vulnerabilities with different utilization targets. Instances highlighted in Blue are Key Registers, Red shapes are Fully Vulnerable (FV) sites, Yellow shapes are partially vulnerable (PV) sites.

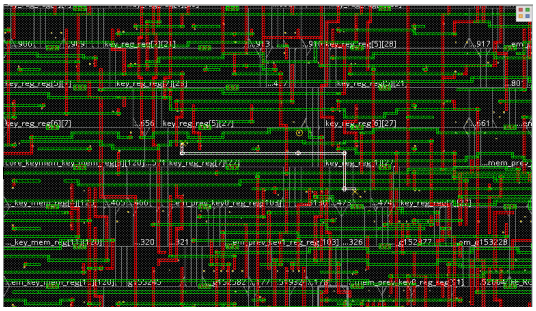
which a large number of routes have been routed are not favorable for adding Trojan logic as performing pin-access to Trojan logic cells and intra-Trojan logic routing will be very difficult to perform without causing DRCs.

When the script is used for evaluation, it highlights the *fully vulnerable (FV)* sites in red. FV sites are sites with enough routing resources over them for allowing pin access and intra-Trojan routing. *Partially vulnerable (PV)* sites are highlighted in yellow. PV are sites where there are a large number of routes, which makes routing difficult for these sites. A text report line 21 in Algorithm 1 is also generated with additional details like the number of routing resources over each vulnerable site and the number of continuous vulnerable regions where the full Trojan (based on the user-specified Trojan size) can be implemented. These areas are determined by calculating the number of vulnerable sites both in the horizontal and vertical directions. These sites are dangerous, as the entire Trojan circuitry can fit inside these regions. The user can use this information to apply any required defense in the layout using the same CAD tool.

In Fig. 4 the security-critical key registers in the design are highlighted in blue. The fully and partially vulnerable sites are highlighted in red and yellow respectively. Another observation from Fig. 4 is that the number of vulnerable placement sites decreases with an increase in utilization. But higher utilization is not always desirable in layouts as



(a) 50% Utilization



(b) 70% Utilization

(c) 90% Utilization

Fig. 5. Visualization of pin access routes in AES Layout with different utilization targets

this makes routing more complex, increases power demand, and causes a large number of timing fails as we will demonstrate in the evaluation section, Section 5. In the AES module as shown in Fig. 4, a large number of vulnerable sites are present in the layout with 50% utilization compare to the layouts with 70% and 90% utilization which has relatively fewer number of vulnerable sites. The increase in cell density increases the net routing density which makes pin access difficult due to limited available routing tracks. Fig. 5 shows the same AES key net and the density of routes in Metal 2 and Metal 3 layers. These layers have thinner widths allowing easier access to pins. The layout with 50% utilization has a large number of unused tracks which makes it easy access to the Trojan logic. The routing density increases for layout with 70% utilization but there are still some unused tracks. In the layout with 90% utilization, the routing density is high and routing to the Trojan logic is difficult and may create shorts or DRC violations.

4.1.2 Distance-based Evaluation and Defense against Trojan Insertion. Increasing the density after the design implementation requires adding logic which causes an increase in power and area. Routing this logic without disturbing existing routes is difficult in complex designs. So to increase density, our method increases the density in a design with 70% utilization with existing logic in the design after the placement stage while allowing enough white spaces in the design for downstream optimizations. Fig. 2 demonstrates our defense using green-colored boxes on the right side of the physical design flow.

The evaluation script Algorithm 1 when used in the flow creates *'gui_shapes'* on the design which indicates the vulnerable areas. The scripts look for these special shapes and increase the density near these vulnerable areas by pulling

existing logic closer together while ensuring that the white spaces are not created in new areas near the security-critical logic. This is done after placement as the tool produces the final placement of cells based on the best PPA requirement. Slight modifications to the design that maintain relatively similar logic placement can have a minimal impact on the PPA, while reducing the number of vulnerable sites. Custom scripting-based fixes allow the designer to increase density locally, just enough to reduce the vulnerable sites in the design without touching the rest of the design. Defensive changes leave a majority of the design open for CAD-based optimization.

To demonstrate distance-based evaluation and defense against the vulnerabilities in the baseline layouts, we attempt to increase the density near the detected vulnerable spaces by our evaluation script in the layout within a specified user-set threshold radius. "Regions" or "Bounds" are created around these areas such that the same logic placed in the logic surrounding these areas are automatically pulled closer to increase the local density. Also, partial blockages are added in the layout within the specified radius so that an even distribution of logic in these regions is done to prevent clustering of logic within a small area leading to a bad redistribution of white spaces in the design. Custom script-based fixes can be done at different stages of the design to increase cell density in the layout based on the white space requirement for each stage as discussed in Section 3.

We demonstrate that we are able to reduce the vulnerability in the layout by a large margin in comparison to the baseline 70% utilization layout with the same area targets by using this approach. We demonstrate that we can reduce the number of fully vulnerable sites and region count close to the values of the design with 90% utilization.

4.1.3 Timing-based Evaluation and Defense against Trojans. The approach described above might not be enough if the setup timing margin on the given path is very large. For example, if a data path has a small number of standard cell stages then the setup margin with which the timing is met on these paths will be a large positive number. If the security critical paths are met by a large positive slack, the radius within which the Trojan logic can be inserted successfully increases drastically. If the positive slack on all security critical paths is little, then the radius around the logic that the user needs to secure will be little and an effective defensive solution can be applied to thwart the attacks. For the lowest possible positive slack, we modify the data path to increase the number of stages without changing the function of the data path by breaking down complex cells into simple logic cells with low drive strength and optimizing them such that timing is satisfied on these paths with a little positive margin.

For evaluation, Algorithm 2 the current positive timing slack on a design that meets timing requirements is determined. Next, the delays of routing and propagation delay of adding an additional NAND gate to the data path are determined by analyzing routing delays and the technology library for standard cells for different output loads and input transition times. Using this data, a pessimistic estimation of the longest possible distance within which a Trojan cell can be inserted while meeting the timing requirements is determined. Using this distance, evaluation similar to distance-based analysis is done to determine vulnerable sites, continuous region count and routing overhead for each vulnerable site.

In timing-based evaluation, it was determined that some security critical paths had a large positive slack resulting in a larger vulnerable radius. To reduce positive slack on these paths, we use the CAD tool attribute of 'dont_touch' on cells. We break down complex logic cells into simple logic cells before placement. This increases the number of stages in the datapath and the extra delays of routing between these cells are added to the datapath. For example, if an OAI (OR-AND-Invert) cell is used in the datapath, we break it down into individual OR, AND, and Inverter cells. The drive strength of these cells is decreased such that the delay from each stage is maximized as the 'dont_touch' setting prevents optimization on the cells with this attribute. Only cells with security-critical logic are affected by this attribute. The tool can optimize the remaining design. We leverage this setting to reduce the positive slack on security-critical paths. A

Algorithm 3: Selective shielding against probing

Input: Security-critical nets and instances SC, Angle Ang
Output: Report $file$

```
1  Function Eval(s, Ang):
2      dir ← {E, W, N, S};                          // Direction East, West, North and South
3      H ← TECHFILE(heights);                       // Get heights of all the metal layers
4      Polygon_Shape ← ∅;
5      foreach h ∈ H do
6          Polygon_Shape[h] ← tan(h, Ang, dir) ;    // Calculate Polygon shape using tan function
7      end
8      Area ← BOOL(Polygon_Shape, s);               // Calculate exposed area
9      return Area
10 Before_Defense ← ∅;
11 foreach s ∈ SC do
12     Before_Defense[s] ← Eval(s, Ang) ;
13 end
14 foreach s ∈ SC do
15     Route s with lower metal layers;             // Narrow width layers
16     Route non-critical around s with higher metal layers;   // Wide width layers
17     Check DRCs ;
18 end
19 After_Defense ← ∅;
20 foreach s ∈ SC do
21     After_Defense[s] ← Eval(s, Ang) ;
22 end
23 file ← {Before_Defense, After_Defense};   // Return the exposed area before and after defense
24 return file
```

lower positive slack ensures that any additional logic cells added to the datapath by an attacker will cause a setup time violation. This reduces the radius within which additional cells can be added. Attempts to remove CAD tool-inserted buffers on the datapaths to increase the positive slack will cause timing violations due to the low drive strength of actual datapath cells. If any of the paths fail to meet timing requirements due to these settings, we identify them, identify the cells which have a maximum delay, and relax the 'dont_touch' attribute before the Clock Tree Synthesis (CTS) stage. The tool then optimizes selected cells till the timing requirements of the path are met with a little positive timing slack. The complete flow is shown in Fig. 4.

4.2 Layout Probing

4.2.1 Assessment. Layout Probing entails determining how vulnerable a design is against electrical and optical probing attacks. The assessment evaluates and reports the exposed area of the standard cells in security critical regions as shown in line 1 of Algorithm 3. This area identifies parts of the standard cells that are visible to an attacker attempting to perform optical probing from the front side of the IC. An attacker can probe targets of interest from various angles and from various sides. The evaluation script considers probe angles of 0, 30, 45, and 60 degrees and probe directions of East, West, North, and South relative to the IC. The probing assessment script accepts the angle of probing as input along with a list of security-critical standard cells. We use the height of the metal layers from the technology file and use this to determine the exposure polygons for the considered angle of attack for the metal layers. Fig. 6 indicates areas of a given instance that are vulnerable to probing for the specified angle of probing.

Algorithm 4: Module level shielding against probing

Input: Security-critical Modules SM, Angle Ang
Output: Report $file$

$Before\_Defense \leftarrow \emptyset$;
foreach $s \in SM$ **do**
  $Before\_Defense[s] \leftarrow Eval(s, Ang)$; // From Algorithm 3
end
$After\_Defense \leftarrow \emptyset$;
$PPA \leftarrow \emptyset$;
$(S\_LEFT, S\_RIGHT) \leftarrow Create()$; // Create left and right shielding cells
foreach $s \in SM$ **do**
  Keep S_LEFT left to s ;
  Keep S_RIGHT left to s ;
  Check DRCs ;
  $PPA[s] \leftarrow Report()$;
  $After\_Defense[s] \leftarrow Eval(s, Ang)$;
end
$file \leftarrow \{Before\_Defense, After\_Defense, PPA\}$; // Return exposed area and overhead
return $file$

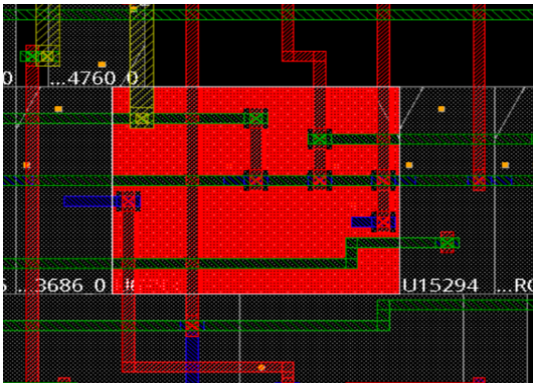
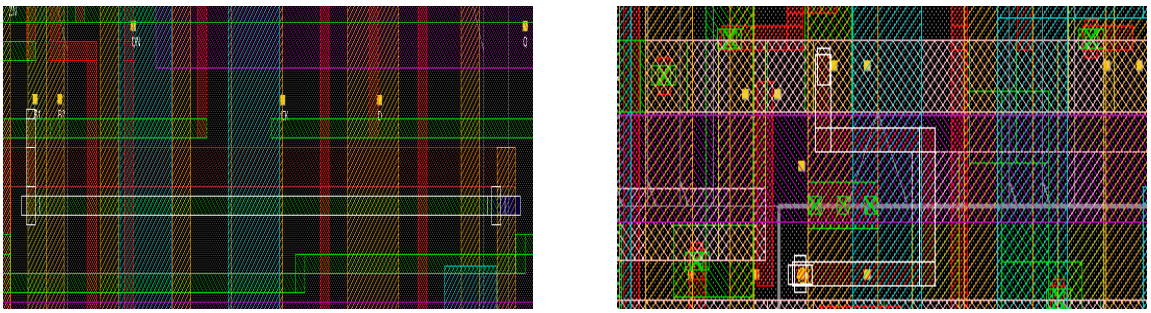


Fig. 6. Visualization of areas (highlighted in Red) of an Instance vulnerable to probing.

For each instance in the list, the script determines the polygons that come under the angle of attack by considering each side (N, S, E, and W) and each metal layer. This is done using basic trigonometric calculations. By using the obtained height of each metal layer from the technology file and the trigonometric "tan" function the polygons of the particular metal layers on the angle of attack are obtained. This process is repeated for all metal layers and a combined polygon list is obtained. The script adjusts each of these polygons to corresponding positions on top of the instance. Boolean AND/OR/NOT operations are performed between the instance polygon and the wire/via polygons to calculate the area of the standard cell that is exposed. The higher the percentage of the standard cell that is exposed, the higher the vulnerability to optical and probing attacks. Similarly, to calculate the exposed area of a sensitive net, we employ a similar strategy. For each segment of the net, we calculate the exposed area and report the percentage of the signal net that is exposed. We verified that the evaluation script works for front-side net probing. We consider angled net probing



(a) Baseline Layout with 70% Utilization

(b) Selective Shielding Defense with 70% utilization

Fig. 7. Security critical net is highlighted by a white border in the layout. Net probing vulnerability for an AES Key net is shown.

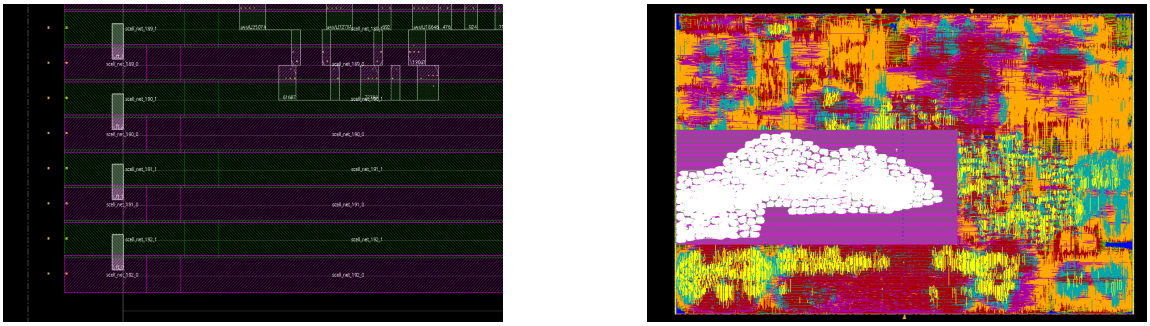
as the future scope and will work on implementing the angled probing analysis for the routes in the different metal layers.

To demonstrate how the layouts can be defended against these types of probing attacks, we explore two different methods of defense. We perform selective instance and net-based shielding from probing attacks and we also perform module-level shielding to fully protect the entire module from the different types of probing attacks. Fig. 2 demonstrates the integration of these defenses into the CAD flow between the CTS and Routing step. That is all the defenses are implemented before routing the regular signal nets of the design.

4.2.2 Selective shielding against probing. Based on the evaluation of the probing vulnerability of AES key registers and their corresponding nets, we created a simple defense approach that takes the security-critical nets and instances as input in Algorithm 3. The security critical nets are routed first with the least amount of distance between the start point and end point using only the lower 2 metal layers. Then all the neighboring design nets which are not in the security-critical nets list are routed over the instance and nets in higher metal layers. The higher-level routes are routed such that they are wider than regular routes for that particular layer. This allows us to reduce the amount of exposure to security-critical instances and nets. This approach provides a selective defense where the non-security critical nets provide the defense to the security-critical nets. The shielding provided by the higher metal layer might be limited by spacing-related DRCs. Fig. 7 shows the difference between the baseline design and a design with selective defense. Both these images show the same net (border highlighted in white) in different layouts. The same net is effectively protected against optical probing in the defensive layout at the higher metal layers as opposed to the baseline implementation.

4.2.3 Module level shielding against probing. For the case of the Module level defense, we attempt to protect the entire module from probing-based attacks, for both instances and nets. This solution is future-proof as it protects the entire cryptographic module from probing-based attacks. So, if any new attack by tapping into different parts of the module other than the key registers and net is designed, this defense protects against those types of attacks as well.

To demonstrate this defense we add two additional layers into the technology LEF file in order to make this possible without any impact on PPA, Algorithm 4. We create our own Shielding cells (S-CELL) which is an anchor for us to create the shielding in the desired area of the layout. There are two types of S-CELL left and S-CELL right. They are kept at the left and right sides of the area to be shielded. Each of these S-CELLs has two pins located in the 2 new metal layers that we have added (metal11 and metal 12). The S-CELL is shown in the module as shown in the Fig. 8. The full



(a) 2 Metal Layers Shielding the specified area

(b) Full module shielding visualization. The area highlighted in white shows actual module placement.

Fig. 8. Module Level defense Implementation.

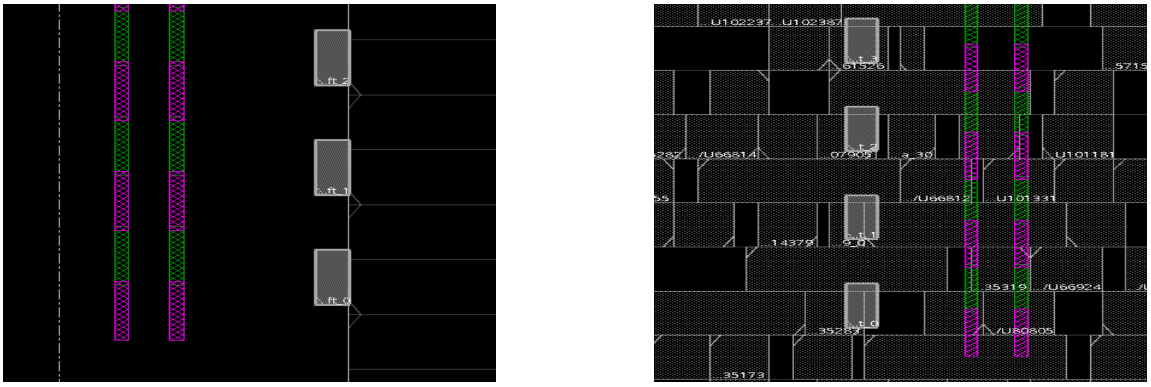
module shielding using 2 additional metal layers is shown. This ensures 0 exposure of the module to probing-based attacks. As the whole shielding process is done using additional layers, this method doesn't have any PPA overheads. If the number of metal layers for a given foundry-based technology LEF is fixed, existing higher metal layers can be leveraged for implementing this defense. But the trade-off here would be that the number of metal layers available for signal routing for this module will be reduced over this area. The routing DRCs and PPA impact of implementing the shield using existing metal layers still need to be explored further and we will work on it in the future to achieve full module-level defense using existing metal layers in the technology LEF.

For the case of module-level defense, the shielding of the security critical module is done just before the routing step in the Place and Route flow. The boundary of the security critical module is obtained from the layout and the S-CELLS are placed on either side of the boundary as shown in Fig. 9 for the entire region height and then the nets are routed between the left and right pins to form the shields based on Non-Default rules (NDR) defined on the added nets that connect the corresponding left and right S-CELLS. The design is routed and goes through post-route optimizations before generating the layout.

5 EVALUATION

5.1 Experimental Setup

5.1.1 Vulnerability of Layout to Trojans. To evaluate our scripts, we use the AES accelerator and a DSP accelerator from the CEP platform, and a standalone OR1200 processor. This is similar to the layouts that were used for evaluation in [14]. Security critical nets are provided as input to the scripts, and the vulnerabilities in the layout are the output generated using evaluation scripts. We consider both the instances and nets of the security critical logic, i.e, the key registers and their output nets for the case of the AES accelerator. Similarly, we consider the *'next_out'* signal and the register from which it is obtained in the DSP accelerators and the *'supv'* bit and its driving register as security critical assets. We consider the most pessimistic case of 20 placement sites for the A2 analog Trojan to evaluate the vulnerabilities in each of these layouts. The designs were synthesized using the Nangate 45nm library [5]. We used the Cadence Genus and Innovus tools for synthesis and place-and-route respectively. But the scripts can easily be ported to different CAD tools. The logic used in the scripts is agnostic to the tools used for obtaining the final layout of the design.



(a) Left side pins for shielding cells (S-CELL).

(b) Right side pins for shielding cells (S-CELL).

Fig. 9. Pins placement for created Shielding cells (S-CELL) for module-level defense.

We used four layouts to evaluate each design. The baseline designs have 50%, 70% and 90% utilization at the floor planning stage. The fourth layout is the layout with our custom defense strategy implementation. This layout also has a utilization of 70% at the floorplan stage. We compare the number of vulnerable sites and continuous region count where the Trojan cells can be inserted in the layout as a metric to evaluate the layout security. The PPA metrics are also compared to evaluate the cost at which the gains in layout defense are achieved. The tool can be used to evaluate the vulnerability of the layout by using 2 evaluation metrics: (1) Distance-based evaluation and (2) Timing-based evaluation.

5.1.2 Vulnerability of Layout to Probing Attacks. When sensitive data needs to be accessed by an attacker, probing attacks pose a significant threat. Therefore, we assess the level of difficulty for an attacker to probe the AES keys in the CEP design. The registers and nets of the AES key registers are evaluated for instance and net probing. That is the area of the instance or the net which are exposed for probing. The case of angled probing is also evaluated for Instance probing vulnerabilities. We evaluate the vulnerability of instance probing attacks from 4 angles: 0, 30, 45, and 60 degrees. Currently, the scripts support front-side probing for net probing. We evaluated the vulnerabilities for 5 CEP layouts. The baseline layouts are the standard CEP layouts with 50%, 70%, and 90% utilization. The fourth layout has additional selective instances and net defenses implemented during the design flow over the 70% utilization layout. Finally, we evaluate vulnerabilities in the layout with module-level shielding.

5.1.3 Vulnerability of Layout to Crosstalk attacks. To show the threat of crosstalk, we modified the routing of some nets similar to [10]. We change the driver strength of the aggressors and victims so that the victim signal can be manipulated. We then increase the frequency of the design and route clock nets closer to the fanout of the 'next_out' signal of the DSP accelerator in the CEP design. We first determine the nets surrounding the security critical logic which are either clock nets or regular signal nets of the DSP accelerator. Assuming the pessimistic case that the attacker can gain control of all signal nets in the vicinity of the victim net, we increase the length of the victim net and routed 12 aggressor nets in parallel to this net on higher metal layers to increase the coupling capacitance between them. Fig. 10 shows the actual routing of these nets to create an attack similar to that in [10].

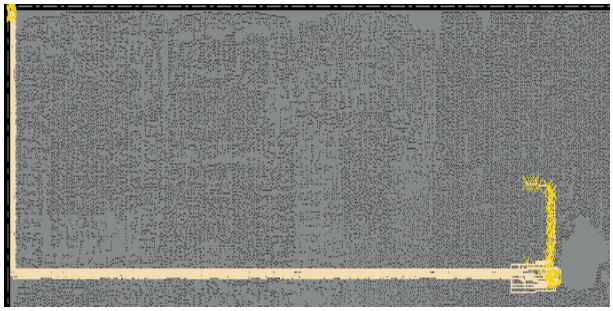


Fig. 10. Routes of 12 parallel nets model the crosstalk attack.

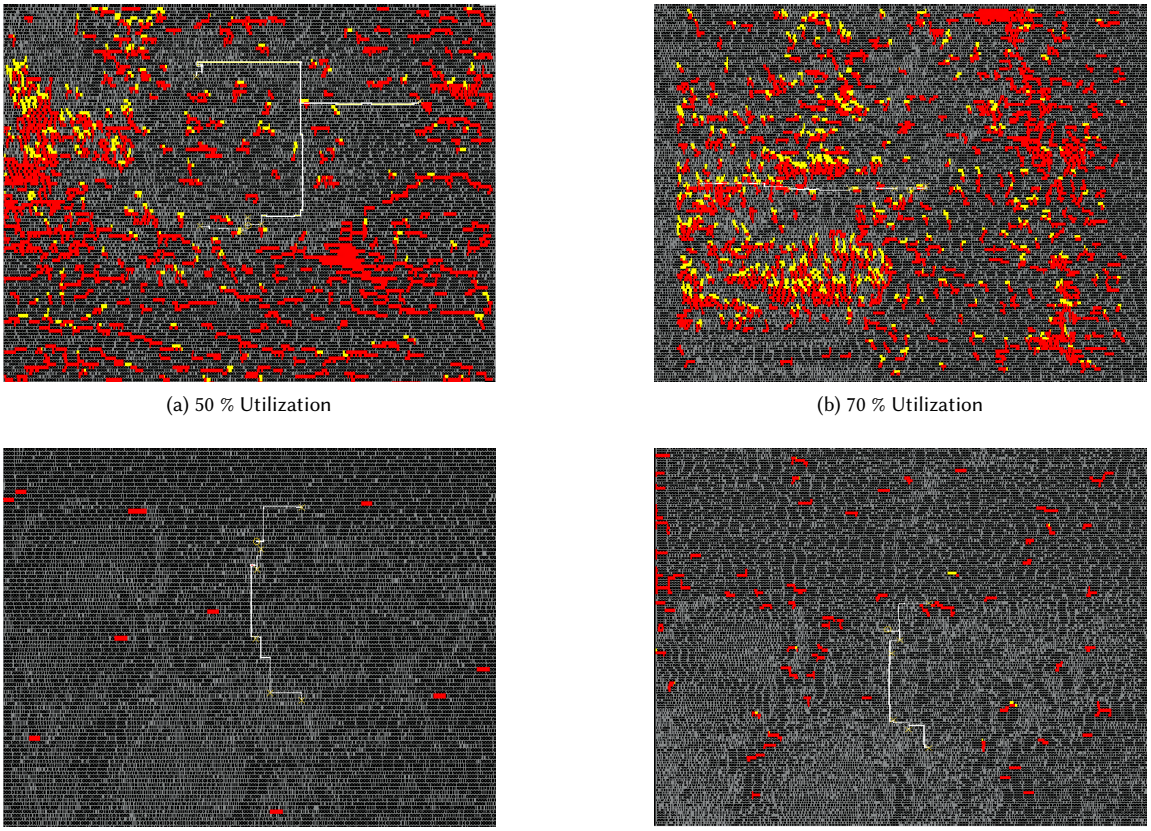
5.2 Results

5.2.1 Evaluation of Vulnerability to Trojans. PPA results along with the layout vulnerability metrics for different utilization targets and the custom defense implementation for the OR1200 layouts are shown in Table 2. Similar results for CEP layouts are shown in Table 4. The frequency for both the CEP and OR1200 was set at 750 MHz. This frequency was chosen after running multiple frequency sweeps to determine the breaking point for both designs at 70% utilization. We use a radius of 25μm around each security critical net and instance to demonstrate our distance-based evaluation for the CEP design. A radius of 100μm is used for the evaluation of the OR1200 design. Both Table 2 and Table 4 show the results of the distance-based evaluation.

From Table 2 notice that increasing the design utilization of the OR1200, decreases the number of vulnerable regions and sites. This reduction comes at the cost of an increase in the number of timing violations. 6000 paths have setup time violations with a large timing slack and this is impossible to fix with standard cell optimization [5]. While layouts with 50% and 70% utilization can complete routing cleanly without DRCs, 90% utilization layout had 12 DRCs. For the custom defense, we meet all PPA requirements as the baseline (70% utilization) while reducing the number of fully vulnerable sites from 3712 $\rightarrow$ 220, decreasing the continuous region count from 722 $\rightarrow$ 28. This defense is comparable to that of layout with 90% utilization. But unlike the 90% utilization design, it meets the PPA requirements without routing DRC violations. Fig. 11 visualizes the layout.

Similarly, Table 4 shows that the layout defense metrics of the custom defense are very close to that of the 90% utilization case while meeting the PPA requirements of the design for the case of AES-based defense. For the case of DSP-based defense, we can achieve better layout defense metrics with our custom defense methodology in comparison to the 90% utilization design using distance-based evaluation. From Table 3 for OR1200 design, timing-based evaluation, 70% utilization has more Vulnerable regions and site count in comparison to 50% utilization layout. This is due to the higher positive timing margin on the path from the security critical logic with 70% utilization. For 90% density, there are no vulnerable regions as the setup time for the security critical path is already failing. If the layout with 90% needs to be sent for fabrication, the frequency of the design needs to be reduced and any remaining DRCs need to be fixed. The performance of the design is defined by the functional specification.

For the case of the AES module in the CEP design from Table 5, the reduction in positive slack helps achieve a more secure layout than the 90% utilization layout. After reducing the positive slack for each individual key bit of the AES module, a total of 2839 fully vulnerable sites across all the key bits combined. This makes implementation of the key leak Trojan which requires 2553 placement sites nearly impossible to implement. The continuous region count is also



(a) 50 % Utilization

(b) 70 % Utilization

(c) 90 % Utilization

(d) Local density increase defense with 70 % Utilization layout

Fig. 11. Vulnerabilities in layout (in red) to Trojans near 'supv' net in OR1200.

much lesser than the 90% utilization design. But due to the much smaller datapath and use of simple logic cells in the case of DSP accelerator path, the positive slack could not be reduced as much as the other designs. But the defense implementation is still much more secure than the baseline 70% utilization layout and comparable to the layout with 90% utilization design.

Table 2. PPA and Vulnerabilities in OR1200 design for layouts with different utilization targets and defenses using a distance-based evaluation.

Layout	***Performance***	***Power (mW)***	***Site count***		***Region***
	WNS;TNS;Viols	***Leak. + Dyn.***	FV	PV	***Count***
50% Utilization	-12 ; -381 ; 77	24.33 + 1045.1	3798	1849	836
70% Utilization	+0.003 ; 0 ; 0	23.54 + 1042.38	3712	544	722
90% Utilization.	-104 ;-184863 ; 6000	23.49 + 1040.9	97	0	10
70% + Defense	0 ; 0 ; 0	23.47 + 1042.1	220	1	28

FV - Fully Vulnerable Sites, PV - Partially Vulnerable Sites

Table 3. PPA and Vulnerabilities in OR1200 design for layouts with different utilization targets and defenses using timing-based evaluation.

Layout	*Performance*	*Power (mW)*	*Site count*		*Region*
	WNS;TNS;Viols	***Leak. + Dyn.***	FV	PV	***Count***
50% Utilization	-12 ; -381 ; 77	24.33 + 1045.1	224	98	12
70% Utilization	+0.003 ; 0 ; 0	23.54 + 1042.38	1801	782	94
90% Utilization	-104 ;-184863 ; 6000	23.49 + 1040.9	0	0	0 (Setup Fail)
70% + Defense	-3 ; -32 ; 25	23.51 + 1043.5	95	0	6

FV - Fully Vulnerable Sites, PV - Partially Vulnerable Sites

Table 4. PPA and Vulnerabilities Comparison in CEP Design for different utilization targets in a defensive layout using a distance-based evaluation.

Layout	*Performance*	*Power (mW)*	*Site count*				*Region Count*	
	WNS;TNS;Viols	***Leak + Dyn***	***AES***		***DSP***		***AES***	***DSP***
			FV	PV	FV	PV		
50% Util.	-30 ; -458 ; 48	34.13 + 1597.7	6036	2822	809	21	2773	25
70% Util.	-1 ; -1 ; 1	33.94 + 1580.23	1821	697	201	18	86	10
90% Util.	-96 ; -4,510 ; 179	33.9 + 1578.56	475	246	166	0	28	9
Def. (AES)	+3 ; 0 ; 0	34.01 + 1580.3	861	474	-	-	51	-
Def. (DSP)	+1 ; 0 ; 0	34.01 + 1580.7	-	-	88	79	-	6

FV - Fully Vulnerable Sites, PV - Partially Vulnerable Sites

Table 5. PPA and Vulnerabilities Comparison in CEP for different utilization targets in a defensive layout using timing-based evaluation.

Layout	*Performance*	*Power (mW)*	*Site count*				*Region Count*	
	WNS;TNS;Viols	***Leak + Dyn***	***AES***		***DSP***		***AES***	***DSP***
			FV	PV	FV	PV		
50% Util.	-30 ; -458 ; 48	34.13 + 1597.7	361891	121773	224453	16598	15675	4213
70% Util.	-1 ; -1 ; 1	33.94 + 1580.23	105312	44705	83322	12154	5709	2665
90% Util.	-96 ; -4,510 ; 179	33.9 + 1578.56	22825	5070	495	19	898	18
Def. (AES)	-15 ; -540 ; 31	33.99 + 1580.7	2839	1869	-	-	201	-
Def. (DSP)	-2 ; -5 ; 4	34.02 + 1580.5	-	-	1351	884	-	142

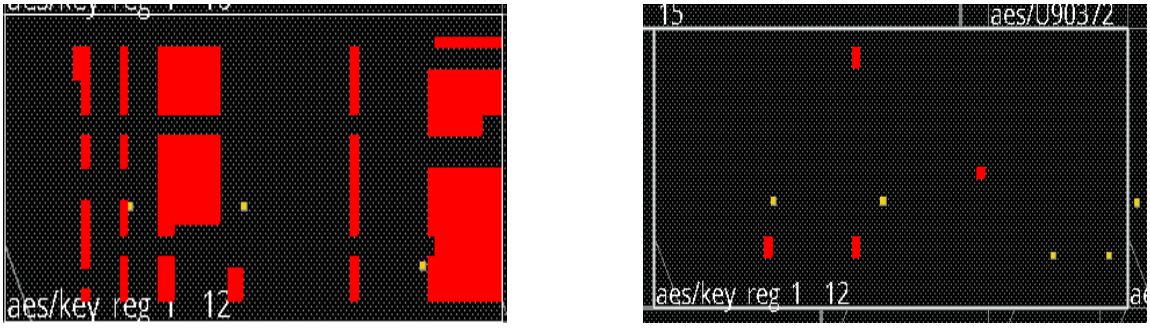
FV - Fully Vulnerable Sites, PV - Partially Vulnerable Sites

5.2.2 Evaluation of Vulnerability to Probing Attacks. Next, we evaluate the susceptibility of layouts to probing-based attacks. The secure logic and nets in the CEP design, such as the AES key registers and their nets are evaluated for the ease with which an attacker can probe from these elements. From Table 6, observe that the baseline layouts without defenses are susceptible to probing. Even though there is a reduction in the average exposure with an increase in utilization, even for a design with 90 % utilization, there is a mean exposure of 10.87% with a maximum value of 58.86% for the AES Key bit nets. Selective shielding brings down the average and maximum exposure to 3.87% and 25.8% respectively with a layout of 70 % utilization. Module-based shielding reduces the exposure to 0% for all key bit nets. Integrating either one of the defenses into the design flow allows layout designers to improve the security of their layouts against net probing.

Similarly, Table 7 provides the evaluation for probing the AES key register instances in the Layout. We consider 4 angles of probing to evaluate instance probing-based attacks, 0, 30, 45, and 60 degrees angled attacks. The average

Table 6. Comparison of Net Probing for AES Key nets

	50% Util.	70% Util.	90% Util.	Selective Shield	Module Shield
Mean	30.9 %	19.39 %	10.15 %	3.87 %	0 %
Median	29.03 %	19.67 %	7.33 %	1.54 %	0 %
Max	92.47 %	74.39 %	58.6 %	25.84 %	0 %



(a) Baseline design with 70 % utilization

(b) Selective Shielding Defense

Fig. 12. Instance probing vulnerabilities of AES Key Register shown in red.

exposure of the AES key register instances decreases from 6.49%→0.8% and 6.15%→0.59% respectively for (0 and 45) degree angle probing. The maximum exposure similarly decreases from 20.06%→4.97% and 32.5%→5.8% respectively for 0 degrees and 45 degrees angled probing for the 70% utilization layout. We observe a similar trend for 30 degrees and 60 degrees angles of probing. The average exposure for all angles of probing decreases from 10% → 6% → 2% when increasing utilization from 50 % → 70 % → 90 %. Selective shielding has a mean value of less than 1% even with a utilization of 70 %. This shows the effectiveness of selective shielding to angled instance probing attacks.

When we integrate probing defenses into the IC layout flow, they have no noticeable differences in PPA in comparison to the baseline. Selective shielding prevents the majority of instance probing attacks as the average exposure for the AES Key bits is less than 1%. The difference between exposure of a register for 70 % exposure versus the same register with selective shielding is shown in Fig. 12. Evaluation for angled instance probing for module-based shielding shows that the layout is secure against all types of probing. Both angled instances probing and net probing are shielded using module-based shielding.

Table 7. Comparison of Angled Instance Probing for AES Key Registers

Angle	50% Util.			70% Util.			90% Util.			Selective Shielding		
	Mean	Median	Max	Mean	Median	Max	Mean	Median	Max	Mean	Median	Max
0	10.09 %	9.14 %	29.63 %	6.49 %	5.94 %	20.06 %	2.07 %	1.27 %	22.44 %	0.80 %	0.49 %	4.97 %
30	10.57 %	9.56 %	35.50 %	6.50 %	5.86 %	28.91 %	2.50 %	1.85 %	20.73%	0.58 %	0.27 %	5.47 %
45	10.07 %	9.23 %	35.10 %	6.15 %	5.58 %	32.50 %	2.20 %	1.55 %	29.30%	0.59 %	0.33 %	5.80 %
60	10.87 %	9.81 %	39.25 %	6.68 %	5.8 %	31.67 %	2.61 %	1.88 %	19.35 %	0.67 %	0.44 %	4.75 %

The Mean, Median and Max values for Module Level Shielding is 0%.

5.2.3 Evaluation of Vulnerability to Crosstalk Attacks. The coupling capacitance increases with an increase in the parallel routing of the nets as shown in Table 8. But due to the increase in density of the design with higher utilization targets, fewer nets can be routed in parallel in the same metal layer. This leads to the routes jumping between different metal layers resulting in lower coupling capacitance. We noticed that the higher density of our defense mechanism hinders the routing of long nets that an external attacker can manipulate to carry out a crosstalk-based attack successfully. Also, after performing the optimization on the datapath to reduce the positive timing slack for timing-based layout defense, increasing the length of the route to 1.2mm results in large setup timing violations of the order of '-5810 ps'. Therefore, it can be concluded that our defensive measures are resilient against crosstalk-based attacks, and any violations in the design's setup will be detected during the post-silicon testing process.

Table 8. Comparison of coupling capacitance.

Layout	***Routing***	***Coupling Cap. (fF)***	
util (%)	***length (mm)***	***Max.***	***Avg.***
50	1.2	107.39	60.46
50	2.7	233.92	148.04
70	1.5	102.74	57.22
90	1.2	31.51	25.98

6 CONCLUSIONS

DEFense scripts scan DEF layouts for vulnerabilities. It presents quantifiable, layout-level security metrics for guidance throughout the physical-design flow. It visualizes threat assessment for interactive designer feedback. We show-case *DEFense* on large-scale MIT LL Common Evaluation Platform system-on-chip design and on the OR1200 processor. We scripted various mitigation strategies during physical design, all the while incurring marginal PPA overheads. We plan to release *DEFense* and case studies to the community. The benefit of applying fixes early in the flow is that it affords degrees of freedom to optimize deterioration in PPA metrics. The study shows the ease with which scripts can be used for evaluation and for defense. Scripting-based fixes can be applied with the standard Physical Design flow to obtain layouts with a high degree of defense with little or no cost to PPA metrics in comparison to baseline designs. We demonstrate that our defenses are robust both in terms of distance-based evaluation and defense and in timing-based evaluation and defense. Distance-based defense demonstrates that increasing the density locally achieves fewer vulnerable sites in a fixed radius. Timing-based evaluation and defenses demonstrate that the effective radius that needs to be secured can be minimized effectively and in turn, the increased density-based defenses can be used to prevent the insertion of Trojans in this smaller radius. Probing-based defenses can be implemented without any significant run time difference and PPA impact. Crosstalk-based attacks seem difficult to implement while our defenses are in place. Evaluation of the defense against Trojan insertion can lead to a runtime increase of ~10%.

REFERENCES

[1] Papa-Sidy Ba, Sophie Dupuis, Manikandan Palanichamy, Marie-Lise Flottes, Giorgio Di Natale, and Bruno Rouzeyre. 2016. Hardware Trust through Layout Filling: A Hardware Trojan Prevention Technique. In *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 254–259. https://doi.org/10.1109/ISVLSI.2016.22

[2] Papa-Sidy Ba, Manikandan Palanichamy, Sophie Dupuis, Marie-Lise Flottes, Giorgio Di Natale, and Bruno Rouzeyre. 2015. Hardware Trojan prevention using layout-level design approach. In *2015 European Conference on Circuit Theory and Design (ECCTD)*. 1–4. https://doi.org/10.1109/ECCTD.2015.7300093

[3] J. Bhasker and Rakesh Chadha. 2009. *Static Timing Analysis for Nanometer Designs: A Practical Approach* (1st ed.). Springer Publishing Company, Incorporated.

[4] Nicole Fern, Shrikant Kulkarni, and Kwang-Ting Tim Cheng. 2015. Hardware Trojans hidden in RTL don't cares — Automated insertion and prevention methodologies. In *2015 IEEE International Test Conference (ITC)*. 1–8. https://doi.org/10.1109/TEST.2015.7342387

[5] FreePDK45 2011. *FreePDK45 TM.* https://eda.ncsu.edu/freepdk/freepdk45/

[6] Christophe Giraud. 2005. DFA on AES. In *Advanced Encryption Standard – AES*, Hans Dobbertin, Vincent Rijmen, and Aleksandra Sowa (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 27–41.

[7] Xiaolong Guo, Raj Gautam Dutta, Yier Jin, Farimah Farahmandi, and Prabhat Mishra. 2015. Pre-silicon security verification and validation: A formal perspective. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–6. https://doi.org/10.1145/2744769.2747939

[8] Aman Gupta. 2018. Hardware Trojan Attack and Defense Techniques. *VLSI and Circuits, Embedded and Hardware Systems Commons* (2018). https://lib.dr.iastate.edu/creativecomponents/391

[9] Samuel T. King, Joseph Tucek, Anthony Cozzie, Chris Grier, Weihang Jiang, and Yuanyuan Zhou. 2008. Designing and Implementing Malicious Hardware. In *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats* (San Francisco, California) *(LEET'08)*. USENIX Association, USA, Article 5, 8 pages.

[10] Christian Kison, Omar Mohamed Awad, Marc Fyrbiak, and Christof Paar. 2019. Security Implications of Intentional Capacitive Crosstalk. *IEEE Transactions on Information Forensics and Security* 14, 12 (2019), 3246–3258. https://doi.org/10.1109/TIFS.2019.2900914

[11] Johann Knechtel, Jayanth Gopinath, Jitendra Bhandari, Mohammed Ashraf, Hussam Amrouch, Shekhar Borkar, Sung-Kyu Lim, Ozgur Sinanoglu, and Ramesh Karri. 2021. Security Closure of Physical Layouts ICCAD Special Session Paper. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. 1–9. https://doi.org/10.1109/ICCAD51958.2021.9643543

[12] Catherine Rooney, Amar Seeam, and Xavier Bellekens. 2018. Creation and Detection of Hardware Trojans Using Non-Invasive Off-The-Shelf Technologies. *Electronics* 7, 7 (2018). https://doi.org/10.3390/electronics7070124

[13] Qihang Shi, Domenic Forte, and Mark M. Tehranipoor. 2017. *Analyzing Circuit Layout to Probing Attack*. Springer International Publishing, Cham, 73–98. https://doi.org/10.1007/978-3-319-49025-0_5

[14] Timothy Trippel, Kang G. Shin, Kevin B. Bush, and Matthew Hicks. 2020. ICAS: an Extensible Framework for Estimating the Susceptibility of IC Layouts to Additive Trojans. In *2020 IEEE Symposium on Security and Privacy (SP)*. 1742–1759. https://doi.org/10.1109/SP40000.2020.00083

[15] Huanyu Wang, Qihang Shi, Adib Nahiyan, Domenic Forte, and Mark M. Tehranipoor. 2020. A Physical Design Flow Against Front-Side Probing Attacks by Internal Shielding. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 10 (2020), 2152–2165. https://doi.org/10.1109/TCAD.2019.2952133

[16] Neil Weste and David Harris. 2010. *CMOS VLSI Design: A Circuits and Systems Perspective* (4th ed.). Addison-Wesley Publishing Company, USA.

[17] Kan Xiao and Mohammed Tehranipoor. 2013. BISA: Built-in self-authentication for preventing hardware Trojan insertion. In *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*. 45–50. https://doi.org/10.1109/HST.2013.6581564

[18] Kaiyuan Yang, Matthew Hicks, Qing Dong, Todd Austin, and Dennis Sylvester. 2016. A2: Analog Malicious Hardware. In *2016 IEEE Symposium on Security and Privacy (SP)*. 18–37. https://doi.org/10.1109/SP.2016.10

[19] Jie Zhang and Qiang Xu. 2013. On hardware Trojan design and implementation at register-transfer level. In *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*. 107–112. https://doi.org/10.1109/HST.2013.6581574

[20] Xuehui Zhang and Mohammad Tehranipoor. 2011. RON: An on-chip ring oscillator network for hardware Trojan detection. In *2011 Design, Automation Test in Europe*. 1–6. https://doi.org/10.1109/DATE.2011.5763260