Benchmarking Advanced Security Closure of Physical Layouts

ISPD 2023 Contest

Mohammad Eslami*
mohammad.eslami@taltech.ee
Tallinn University of Technology
Estonia

Johann Knechtel*
johann@nyu.edu
New York University Abu Dhabi
UAE

Ozgur Sinanoglu
ozgursin@nyu.edu
New York University Abu Dhabi
UAE

Ramesh Karri
rkarri@nyu.edu
New York University
USA

Samuel Pagliarini
samuel.pagliarini@taltech.ee
Tallinn University of Technology
Estonia

ABSTRACT

Computer-aided design (CAD) tools traditionally optimize "only" for power, performance, and area (PPA). However, given the wide range of hardware-security threats that have emerged, future CAD flows must also incorporate techniques for designing secure and trustworthy integrated circuits (ICs). This is because threats that are not addressed during design time will inevitably be exploited in the field, where system vulnerabilities induced by ICs are almost impossible to fix. However, there is currently little experience for designing secure ICs within the CAD community.

This contest seeks to actively engage with the community to close this gap. The theme is security closure of physical layouts, that is, hardening the physical layouts at design time against threats that are executed post-design time. Acting as security engineers, contest participants will proactively analyse and fix the vulnerabilities of benchmark layouts in a blue-team approach. Benchmarks and submissions are based on the generic DEF format and related files.

This contest is focused on the threat of Trojans, with challenging aspects for physical design in general and for hindering Trojan insertion in particular. For one, layouts are based on the ASAP7 library and rules are strict, e.g., no DRC issues and no timing violations are allowed at all. In the alpha/qualifying round, submissions are evaluated using first-order metrics focused on exploitable placement and routing resources, whereas in the final round, submissions are thoroughly evaluated (red-teamed) through actual insertion of different Trojans.

CCS CONCEPTS

• **Security and privacy → Security in hardware**; • **Hardware → Physical design (EDA)**.

KEYWORDS

Hardware Security; Physical Design; Security Closure; Contest; Hardware Trojans; ASAP7;

*Both authors contributed equally to this research.

ACM Reference Format:
Mohammad Eslami, Johann Knechtel, Ozgur Sinanoglu, Ramesh Karri, and Samuel Pagliarini. 2023. Benchmarking Advanced Security Closure of Physical Layouts: ISPD 2023 Contest. In *Proceedings of the 2023 International Symposium on Physical Design (ISPD '23), March 26–29, 2023, Virtual Event, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3569052.3578924

1 INTRODUCTION

This paper presents the second contest on *security closure* of physical layouts, i.e., on the challenge of hardening the physical layouts of integrated circuits (ICs) at design time against hardware-security threats that are executed post-design time.

Bringing this topic to the physical-design community is important for multiple reasons. First, many threats for hardware security, like Trojan insertion or side-channel attacks, are directly targeting for vulnerabilities of the physical layouts. Second, threats that are not mitigated during design-time are almost impossible to fix later on; ICs are unlike patchable software. Third, even if efforts are taken toward secure design at higher abstraction layers (like logic synthesis), such efforts may be undermined later on by, e.g., power, performance, and area (PPA) optimization, thus becoming futile without dedicated support for security closure at layout level.

This paper is organized as follows. We outline the theme, general approach, and some logistics in this Sec. 1. In Sec. 2, we discuss the scope and background for the contest and outline tasks as well as possible directions for solving them. In Sec. 3 we describe the implementation and evaluation of the contest in detail. The contest website [15] provides further information; importantly, all benchmarks and results will remain online there after the contest concludes, to stimulate further interest from the community.

1.1 Theme and Context

Securing the omnipresent information technology is an important but tough endeavour that requires efforts all the way from software down to the hardware. For the design, manufacturing, and deployment of ICs, there are numerous companies and partners involved within complex and world-wide supply chains. ICs run through many hands, where some of those may be acting with malicious intent. Furthermore, once ICs are deployed in the field, an even larger attack surface arises. Most relevant for the physical-design community is that computer-aided design (CAD) tools traditionally optimize "only" for PPA, whereas modern CAD flows should also incorporate techniques for secure IC design. See also, e.g., [9, 13, 16, 17] for further reading.

This contest is part of the International Symposium on Physical Design (ISPD) 2023. Participants will focus on securing the physical layout of ICs. Acting as security engineers, participants will iteratively and proactively evaluate and fix the vulnerabilities of IC layouts at design-time.

The threat to consider in this contest—Trojan insertion—represents a scenario with clear relation to physical design for defending against. Further, the contest scope is well constrained, thereby easing the ramp-up for participants new to hardware security.

1.2 Objective and General Approach

The objective of this contest is the following. Implement physical-design measures to proactively harden layouts against post-design Trojan insertion, conducted during mask generation or manufacturing. See Sec. 2.1 for more details on this threat scenario.

To achieve this objective, participants would want to, e.g., revise placement and routing such that insertion of Trojan trigger components as well as routing trigger and payload components becomes difficult, all while accounting for the impact on design checks and PPA by the defense approach. Given that there are different, possibly competing metrics to be considered for design quality and security closure at once, some machine learning-based guidance could be promising here. In any case, there is no single, right or wrong approach toward that end—it is up to the participants' creativity and skills to come up with the best defense schemes.

Participants can work on any physical-design platform of their choice, be it commercial tools, prominent open-source tools like *OpenROAD* [11], or custom in-house tools.[1] In any case, before devising or even implementing some defense measures, participants would want to i) understand the scope in general and the threat in particular (Sec. 2), ii) understand the way the threat is considered and scored for this contest (Sec. 3.4), and iii) be as creative as possible for security closure while not "re-inventing the wheel" for core CAD algorithms and design techniques.

1.3 Logistics

This contest is open to students of all levels (undergrads, graduates, and/or post-graduates) as well as practitioners from industry, with prizes limited to academic participants.

The benchmarks are physical layouts of various crypto cores. We provide all relevant files along with the layouts. See Sec. 3.3 and the contest website [15] for more details.

The scoring employs a weighted function considering security metrics as well as design-quality metrics. There are also constraints to be considered, importantly that no design rule check (DRC) and no timing violations are allowed. See Sec. 3.4 for more details.

There is an alpha/qualifying round, where we provide a set of benchmarks early on. (Even before that, we release some sample benchmarks.) All participants that submit, for each benchmark, some valid solution(s) providing any improvement for the overall score, move on to the final round. There, we ramp up the challenge; while the alpha round considers only first-order metrics for Trojan insertion, the final round considers actual Trojan insertion for more thorough and realistic assessment of the security of the layouts submitted by participants. During both the alpha and final rounds, results and current rankings are shared regularly for those participants who opt in; this is meant to spur the contest throughout the whole timeline which spans several months.

Participants can interact with the organizers through a dedicated mailing list. We also publish questions and answers (Q&As) regularly on the website [15].

The final results, rankings, and awards will be first announced at ISPD and then published on the contest website [15] as well. Cash prizes and award plaques will be given to the top three teams. Top teams are encouraged to disseminate their results and means for security closure further with the community, but that is not a requirement for participation.

[1]However, the use of commercial tools, in particular Cadence Innovus, is recommended for quicker ramp-up. We have implemented and thoroughly evaluated a related reference design flow (see Sec. 3.3.3, also including modifications as needed for the library of choice, ASAP7 [4] (see Sec. 3.2).

2 BACKGROUND

2.1 Hardware Security

2.1.1 Overview. Due to the changes in the business model of manufacturing ICs, it is rarely the case anymore that circuits are designed and fabricated by the same entity [23]; companies outsource the fabrication process to third parties. This outsourcing brings new challenges to the security and trustworthiness of ICs since there is significantly less control and oversight during the fabrication process.

The main threats during the fabrication stage include Trojan insertion, IP piracy, and illegal overproduction [5, 19, 22]. For example, IP piracy refers to copying and illegal sale or reuse of intellectual design property extracted from the chip during fabrication. For hardware security in general, there is a wide range of other threats as well, including physical attacks to retrieve sensitive information in the field [13].

Hardware Trojan is a generic term for malicious modifications to a design, either via addition, subtraction, or replacement of the existing logic. A rogue engineer in a concerned third-party company could, potentially, manipulate the intended behavior of a chip, either in parts or in the whole batch of the production line (Fig. 1). The malicious intent might be to disrupt functionality, leak information, damage the chip, or reduce the performance by increasing the power consumption or temperature, etc. [12].

2.1.2 Hardware Trojans. As indicated, a Hardware Trojan may leak sensitive information (i.e. secret key used in a crypto core) or change the original behavior of the design so that it could destroy the chip during operation. An activation mechanism, also called trigger, is typically based on specific and rare combinational and/or sequential conditions. Once the trigger condition is met, the Trojan main's circuitry, also called payload, performs its malicious operation.

In case a Trojan has a significant impact on the PPA or/and on the functionality, or in case its size is relatively large, it might be easy to detected. Hence, an adversary should smartly design and insert Trojans so that they remain hidden during testing and normal operation [10, 27].

Hardware Trojans are difficult to identify using traditional post-manufacturing testing that uses functional, structural, or random

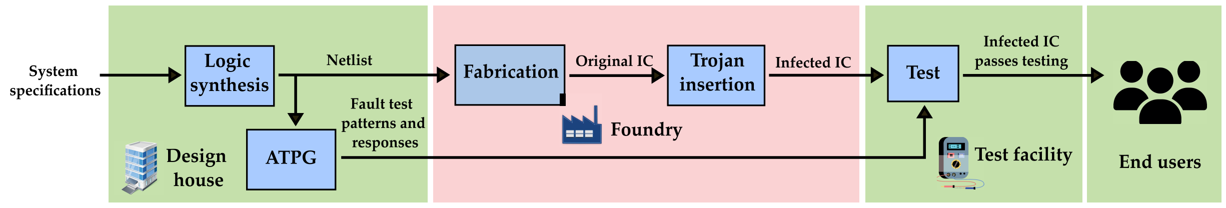


Figure 1: The simplified IC supply chain from design to end-user, including the threat model for this contest. The foundry is considered untrusted, thus marked red; an adversary would insert some Trojan during fabrication itself. Reused from [18] with permission.

patterns [7, 26]. This is because the generation and application of manufacturing tests aim to find flaws or unacceptably wide changes in device parameters that lead to divergence from functional or parametric specifications. Such tests are, however, not well suited to determine malicious but rare, additional functionality brought on by Trojans or to determine alterations in circuit behavior brought on by random unusual events. When seeking to adopting a post-silicon test/validation method to reliably detect Trojans, there are a number of significant hurdles.[2] Thus, other approaches like design-time security closure against Trojan insertion are important [25].

2.1.3 Security Closure Against Hardware Trojans. As outlined, Trojan detection and diagnostic approaches have certain limitations, including the identification of uncommon nodes, process variation, and measurement error. Thus, ICs should be devised with some self-protection awareness if these methods are to be made more effective. The primary modes for Trojan prevention, at the moment, include obfuscation, layout filling, split manufacturing, and insertion of self-testing circuitry [20, 25, 26]. Techniques for bridging layout gaps with functional logic are suggested to make Trojan detection easier and decrease the possibility of Trojan insertion [25, 26].

2.2 Threat Model

For this year's contest, we consider Hardware Trojans inserted at the post-layout stage as the threat to be tackled. Adversaries have access to all technology details and cell libraries used by the victim to create the layout because they are familiar with the foundry's manufacturing process. However, we assume that adversaries are unaware of the specific timing/power limitations, clock domains, input/output pin functionality, or high-level functionality of the victim's design, while they are knowledgeable about IC design and have access to contemporary EDA tools.

This threat model is compatible with state-of-the-art work, in particular with recently demonstrated Trojan insertion in actual silicon [21].

[2]First, an adversary can come up with a large number of possible Trojan types and realize excessively many different versions of all shapes and sizes. Thus, Trojans differ greatly in terms of their structural and functional characteristics, and also because of their covert nature, it can be very difficult to activate some unknown Trojans and observe their results. Therefore, deterministic and even exhaustive testing methods seem to be impractical. Process variation and measurement noise also present themselves as significant obstacles to observing Trojan effects in physical parameters, such as delay and supply current. Increased process variation in advanced technologies in particular can conceal Trojan-related effects in physical parameters.

3 IMPLEMENTATION AND EVALUATION

3.1 Platform

3.1.1 Tool Flow for Participants. Recall that participants are free to use any physical-design tools of their choice, be they commercial, open-source, or in-house tools. Still, as this is the second contest in a row, it is more advanced and demanding, also because the ASAP7 library is used (see Sec. 3.2). Thus, we recommend that participants employ the suggested tools and the provided reference design flow for quicker ramp-up.

3.1.2 Backend for Organizers. The evaluation backend is based on commercial design tools, including Cadence Genus, Innovus, Conformal, etc. The actual evaluation, the parsing of reports, the computation of metrics and scores, and the file management tasks are all implemented using *tcl* and *bash* scripting.

The backend is implemented as a daemon. It supports parallel processing of various submissions from different teams, and further supports parallel processing of all calls to commercial tools for individual submissions. This implementation approach significantly reduces the runtime. The workflow is as follows:

(1) Initialize. Global runtime variables, like all the uniform resource locators (URLs) for the participants' private submissions sites are retrieved and memorized, enabling faster access later on. Local work and backup folders are initialized. This step supports 'testing' versus 'production' modes; the daemon is run separately twice on our server to support both modes in parallel.
(2) Download of new submissions. Any new submissions are downloaded and queued for evaluation, considering the current overall workload of the backend and the currently ongoing runs (if any) of the concerned participants. For fairness, all participating teams are given some upper limits for parallel processing.
(3) Evaluation. Once some submissions pass the queue, evaluation is started, and an email notification is sent to the concerned participants. The evaluation itself is conducted in multiple steps, with basic design checks done first and more complicated evaluation next and in parallel. All these valuation steps are implemented in *tcl* and *bash* scripts and can be easily updated, without the need to revise the backend daemon itself.
(4) Upload of new results. Once the evaluation is done, a follow-up email notification is sent, which also contains the scores and important messages, like current processing status (for

all other submissions) and errors or warnings observed for the particular submission.

Steps (2)–(4) are repeated periodically and automatically by the daemon. All daemon procedures also feature status monitoring and logging for robust processing. An exemplary screenshot of the daemon is shown in Fig. 2.

While specifically developed and tailored for this contest, the backend scripts are also written with some flexibility in mind. We are releasing all backend scripts at [14] as these can be helpful to the community for implementation of other contests.

We did not release the backend scripts to the participants during the contest itself. Doing so would not have provided any benefit to the participants, as all the important evaluation and scoring scripts are already released along with the benchmarks. Thus, participants are able to run all important scripts independently at their end, to help implementing and debugging their ideas.

3.1.3 Frontend for Participants. With the outlined workflow for the backend, we require some frontend for exchange of submission and result files. With reliability and world-wide availability in mind, we opt for *Google Drive* as web frontend.

All registered teams are provided access to their dedicated Google Drive folder. Teams may upload submission files anytime, upon which new files are automatically downloaded by our backend for evaluation. Results are returned into the teams' respective benchmark folders. Results will include the overall score but also report and log files as generated by our backend, to provide participants with more detailed insights.

3.2 ASAP7 PDK and Library

One of the significant changes in the present edition of the contest is the adoption of a PDK that is much more modern. The ASAP7 PDK [6], originally developed by the team from Arizona State and ARM, is likely the most complete PDK developed by academia. The same team also provides a standard cell library [24] for their PDK, one that utilizes FinFET transistors and is properly characterized. The many files provided do resemble a commercial PDK, including multi-Vth cells, extraction decks, DRC decks, and so on.

For this contest, a few modifications were made to the library, mainly to ensure that participants could use different physical-synthesis tools and versions with ease. Some details are given next.

A few complex via rules were dropped, while still maintaining the major features of the technology. In tandem, we have also added new design rules that were not part of the original technology setup; we added those to create interesting and challenging scenarios for the participants to work around. One of the most significant addition has been the notion of colored metals, i.e., metal layers that would be fabricated using more than one mask. This departure from the original setup of the ASAP7 PDK introduces some challenges that are common in the first generation of FinFET technologies.[3] We have also introduced max density rules for all metal layers; this is meant to discourage the participants from adopting some simple shield-based solutions to protect their layouts.

More details can be obtained from the technology LEF file provided in the benchmarks release [15], or directly from our repository containing the modified ASAP7 technology and the reference design flow [4]. Note that all significant changes are annotated as comments.

[3]In more mature technologies, where two metal shapes of the same layer are drawn side by side, the minimum distance between them depends on the width and parallel-run length of the shapes. In technologies where coloring is considered, the minimum distance changes also depending on whether the metals are in the same color or in different colors.

3.3 Benchmarks

3.3.1 Overview. We consider 6 different crypto cores as benchmarks: AES128, Camellia, CAST, MISTY, SEED, and SHA256. These cores have different sizes and complexities, ensuring different difficulty levels across the benchmarks.

For all benchmarks, we first pass the RTL description [1–3] to the logic synthesis tool, Cadence Genus, followed by the physical implementation using Cadence Innovus. For each benchmark, we use custom timing constraints while the same ASAP7 library is used for all benchmarks. For logical and physical synthesis, we have refrained from utilizing optimization on purpose, namely to keep some margin for the participants to work with. In other words, we enable participants to explore trade-offs between security and design metrics. The vanilla scripts for logical and physical synthesis have been made available early on [4, 15], to help the participants adapt to the ASAP7 library. More details for the physical design are provided in Sec. 3.3.3.

The benchmark release includes the post-route Verilog netlist and DEF file. It also contains the design database, the SDC timing files, the reports for the evaluation and scoring of the baseline layout, the evaluation and scoring scripts, and list of *cell assets.*

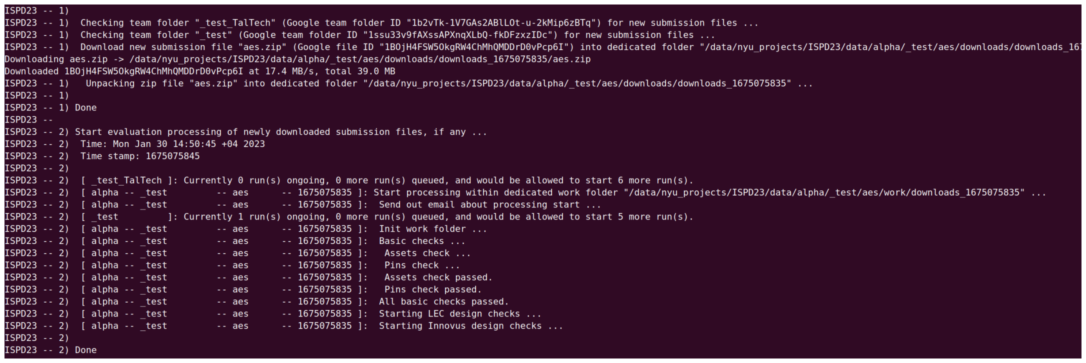
Cell assets are selected manually from all flip-flops (FFs) to represent potential locations, like key registers, that some Trojan could connect to for its trigger and/or payload. Note that, for the alpha round, participants are informed about all the assets, but there is no specific evaluation or scoring related to assets. In the final round, specific Trojans will be targeting at subsets of those assets. In any case, all assets must be maintained by participants.

3.3.2 Sample Benchmark: SHA256. Early on, a sample benchmark, SHA256, was provided for the participants as a warm-up design to help them get familiar with the ASAP7 library as well as adapt to strict rules and constraints of the contest. We include this sample benchmark along with the other benchmarks for both alpha and final rounds since the same strategy and implementation flow are used for the sample and the final benchmarks.

Fig. 3 shows layout details for the SHA256 benchmark.

3.3.3 Implementation Flow. Next, we provide more details about the physical implementation. The script for the sample benchmark SHA256 is made available early on at [4]. It can be used by participants for other benchmarks, requiring only minor customizations for different designs. This script includes the following steps:

(1) **Defining Globals:** Global variables like the version of the tool used, the design technology, and the number of available processor cores are set, along with all the paths for the initial netlist, libraries, LEF files, and timing constraint files. Participants are not allowed to use different library files.



```
ISPD23 -- 1)
ISPD23 -- 1)  Checking team folder "_test_TalTech" (Google team folder ID "1b2vTk-1V7GAs2ABlLOt-u-2kMip6zBTq") for new submission files ...
ISPD23 -- 1)  Checking team folder "_test" (Google team folder ID "1ssu33v9fAXssAPXnqXLbQ-fkDFzxzIDc") for new submission files ...
ISPD23 -- 1)  Download new submission file "aes.zip" (Google file ID "1BOjH4FSW5OkgRW4ChMhQMDDrD0vPcp6I") into dedicated folder "/data/nyu_projects/ISPD23/data/alpha/_test/aes/downloads/downloads_167
Downloading aes.zip -> /data/nyu_projects/ISPD23/data/alpha/_test/aes/downloads/downloads_1675075835/aes.zip
Downloaded 1BOjH4FSW5OkgRW4ChMhQMDDrD0vPcp6I at 17.4 MB/s, total 39.0 MB
ISPD23 -- 1)   Unpacking zip file "aes.zip" into dedicated folder "/data/nyu_projects/ISPD23/data/alpha/_test/aes/downloads/downloads_1675075835" ...
ISPD23 -- 1)
ISPD23 -- 1) Done
ISPD23 --
ISPD23 -- 2) Start evaluation processing of newly downloaded submission files, if any ...
ISPD23 -- 2)  Time: Mon Jan 30 14:50:45 +04 2023
ISPD23 -- 2)  Time stamp: 1675075845
ISPD23 -- 2)
ISPD23 -- 2)  [ _test_TalTech ]: Currently 0 run(s) ongoing, 0 more run(s) queued, and would be allowed to start 6 more run(s).
ISPD23 -- 2)  [ alpha -- _test         -- aes      -- 1675075835 ]: Start processing within dedicated work folder "/data/nyu_projects/ISPD23/data/alpha/_test/aes/work/downloads_1675075835" ...
ISPD23 -- 2)  [ alpha -- _test         -- aes      -- 1675075835 ]:  Send out email about processing start ...
ISPD23 -- 2)  [ _test         ]: Currently 1 run(s) ongoing, 0 more run(s) queued, and would be allowed to start 5 more run(s).
ISPD23 -- 2)  [ alpha -- _test         -- aes      -- 1675075835 ]:  Init work folder ...
ISPD23 -- 2)  [ alpha -- _test         -- aes      -- 1675075835 ]:  Basic checks ...
ISPD23 -- 2)  [ alpha -- _test         -- aes      -- 1675075835 ]:   Assets check ...
ISPD23 -- 2)  [ alpha -- _test         -- aes      -- 1675075835 ]:   Pins check ...
ISPD23 -- 2)  [ alpha -- _test         -- aes      -- 1675075835 ]:   Assets check passed.
ISPD23 -- 2)  [ alpha -- _test         -- aes      -- 1675075835 ]:   Pins check passed.
ISPD23 -- 2)  [ alpha -- _test         -- aes      -- 1675075835 ]:  All basic checks passed.
ISPD23 -- 2)  [ alpha -- _test         -- aes      -- 1675075835 ]:  Starting LEC design checks ...
ISPD23 -- 2)  [ alpha -- _test         -- aes      -- 1675075835 ]:  Starting Innovus design checks ...
ISPD23 -- 2)
ISPD23 -- 2) Done
```

Figure 2: Screenshot of the backend daemon working in ‘testing’ mode. Listed is the download, initial design checks, and start of commercial tools for detailed evaluation, all for one submission of the AES128 benchmark.

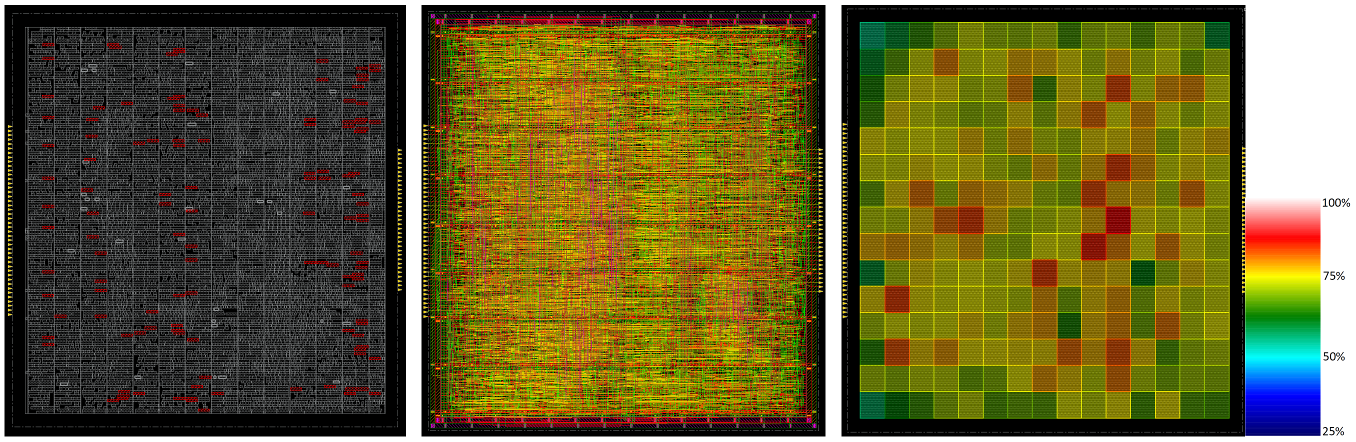


Figure 3: Layout details for the SHA256 benchmark. (Left) placement with cell assets highlighted in red. Note that input pins are placed on the left side, whereas output pins are placed on the right side. (Middle) routing layers. (Right) cell density map.

(2) **Floorplanning:** The size of the floorplan should be defined properly based on which benchmark is being implemented. Participants are free to define the floorplan and aspect ratio. At the same time, power planning (see also further below) has to be accounted for, including parameters for ring spacing, ring offset, ring size, stripe frequency and stripe-to-stripe distance. These parameters are fixed for all designs; participants have to maintain the same floorplan/powerplan strategy for fairness.

(3) **Pin Assignment:** The location of IO pins can influence the quality of the design. Thus, we place and constrain all input pins to the left side of the designs and all the output pins on the right side.

(4) **Power Distribution Network:** The ASAP7 PDK and library are rather restrictive regarding how power stripes can be defined. There are only a few combinations of metal layers, width, spacing, and offset that can generate a coherent power network with adequate via arrays. Taking this into account, the core rings are specified to be routed using M6 and M7 metal layers. For the standard-cell rails, follow pins appear in both M1 and M2 in what is called “stapled style.” Finally, the vertical and horizontal stripes are specified to be routed using M3 and M4 metal layers, respectively. Once all these parameters are set, the power distribution network (PDN) is routed and power vias are generated. Participants should not modify this power distribution strategy, except for adjusting it to smaller/larger floorplans.

(5) **Place and Route:** First, standard cells are placed in the core area. If the floorplan is too small to fit all standard cells, participants should revisit floorplanning and resize the floorplan accordingly. Once the placement of standard cells is passed, the clock is ready to be distributed (clock tree synthesis, CTS). After performing CTS, the design is ready to be routed. Routing is one of the last steps in physical design and takes typically the largest share of the implementation time. Note that, after routing, it is very likely that some violations occur, due to different reasons. Solving these violations, especially those related to pin access, which can become very challenging for dense layouts, is also part of the challenge put forward in this contest.

(6) **Generating Reports, Exporting Final Layout:** Once the design passed all necessary checks and verification, it is ready

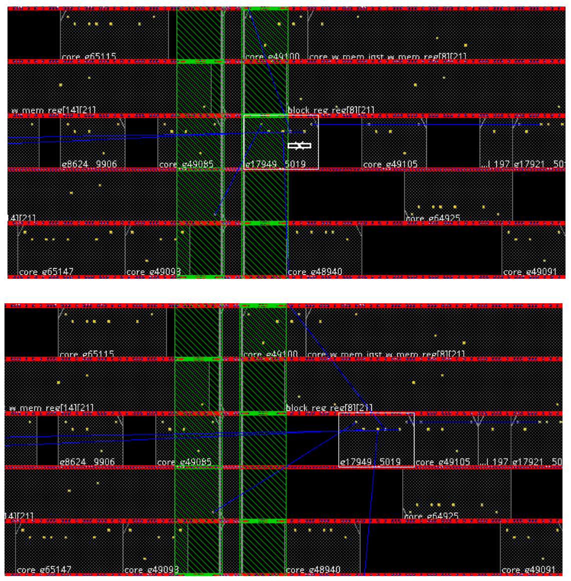


Figure 4: An example for fixing pin access DRC violations. (Top) DRC violation around a power stripe. (Bottom) Fixing the violation by moving some instances.

to be exported. Furthermore, post-route reports for PPA, etc., can be generated. Participants are required to generate DEF files and post-route netlist files for submission.

As indicated, with the reference design flow [4], some DRC violations are *expected* especially for pin access around the power stripes. These violations can be easily fixed manually, by moving the standard cells away from the power stripes (Fig. 4). For larger designs, a semi-automated approach to detect and fix these violations might be devised by the participants.

3.3.4 Alpha, Final Rounds Benchmarks. After the warm-up phase, 5 crypto cores (AES128, Camellia, CAST, MISTY, and SEED) are added to the sample benchmark (SHA256). As indicated, the implementation flow for logical and physical synthesis of these benchmarks is the same as for the sample benchmarks. The only differences are using specific timing constraints and different floorplan sizes for each design.

As mentioned before, the benchmarks exhibit different levels of complexity, size, and density; benchmarks can be classified into categories from 'easy' to 'difficult.' The cell density maps for selected benchmarks are shown in Fig. 5. There, red marks high-density areas, while green and blue marks low-density areas. Note that the layouts underlying for Fig. 5 are not in the same scale; thus, different grid sizes are used for comparable representation. Details for layout dimensions and grid sizes are given in Table 1.

Another parameter/metric for the layout complexity is routing congestion and utilization in each metal layer. Routed layouts are shown in Fig. 6. For example, for AES128 (left-most subfigure), more pink areas indicate that this required is more utilized as it required more routing within the top metal layer (pink = M7).

Table 1: Specification of the benchmark layouts

Benchmarks	Dimensions (µm)	Density Grid (# Rows)
AES128	822.44×822.44	14×14
Camellia	158.24×158.24	8×8
CAST	293.24×293.24	14×14
MISTY	174.44×174.44	10×10
SEED	206.84×206.84	10×10
SHA256	190.64×190.64	11×11

3.4 Metrics and Scoring

For security evaluation, we consider two sets of metrics and phases for scoring this year. In the first phase, i.e., for the alpha/qualifying round, submissions are evaluated using first-order metrics. In the second phase, i.e., for the final round, we extend these simple metrics with results for actual Trojan insertion.

3.4.1 Design Metrics, First-Order Security Metrics (Alpha Round). For evaluating the quality of the design, PPA metrics (power; worst negative slack, WNS; area) are considered. For evaluating the resilience, security metrics describe the layout resources remaining for Trojan insertion. (Thus, participants should reduce unused resources, i.e., open placement sites and free routing tracks, as much as possible for better scoring.) Both security and PPA metrics are evaluated over the baseline to obtain the scoring.

Next, the metrics are categorized.

(1) Security – *sec*
  (a) Trojan insertion – *sec_ti*
    (i) Placement sites of exploitable regions (ers)
      - Max # of sites across all ers – *sec_ti_sts_max*
      - Median # of sites across all ers – *sec_ti_sts_med*
      - Total # of sites across all ers – *sec_ti_sts_sum*
    (ii) Routing resources of exploitable regions (ers)
      - Total # of free tracks across all ers – *sec_ti_fts_sum*
      - Note that, for each exploitable region, free tracks are summed up across all metal layers.
(2) Design quality – *des*
  (a) Power
    - Total power – *des_pwr_tot*
  (b) Performance
    - Worst neg. slack, setup timing req. – *des_prf_WNS_set*
    - Worst neg. slack, hold timing req. – *des_prf_WNS_hld*
  (c) Area
    - Total die area (*not* standard cell area) – *des_ara_die*

3.4.2 Actual Trojan Insertion (Final Round). For each design, we attempt actual insertion of different Trojans. The possible outcomes—from the participant's perspective as defenders—would be 'fail' if the Trojan insertion is successful, 'partial pass' if the insertion succeeds but does compromise timing of the design, and 'full pass' if the insertion fails, e.g., induces some DRC violations. Note that this scoring can be further augmented with other metrics.

For the task of Trojan insertion, we use an ECO-based flow similar to that proposed in [8]. We use three different types of Trojans that: (i) leak information, (ii) modify the output value of FFs, and (iii) over-consume power. For the first two types, the target is chosen from the cell assets. The third type can be connected to

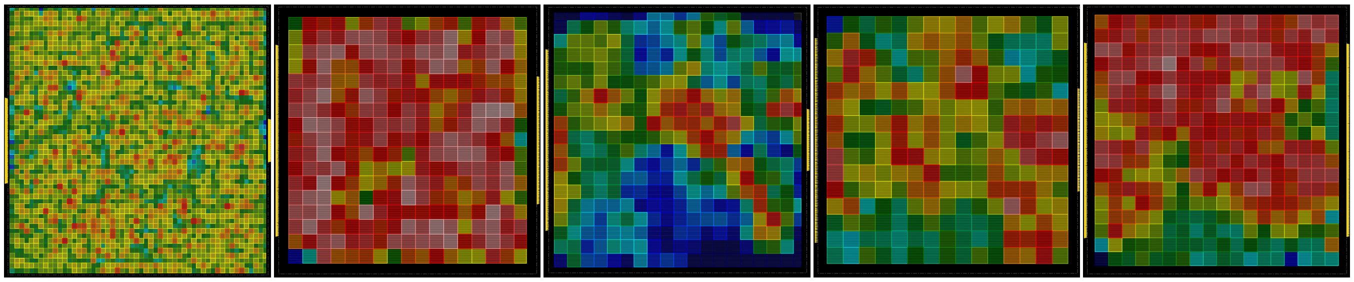


Figure 5: Cell density maps for AES128, Camellia, CAST, MISTY, and SEED (left to right). See Fig. 3 for legend.

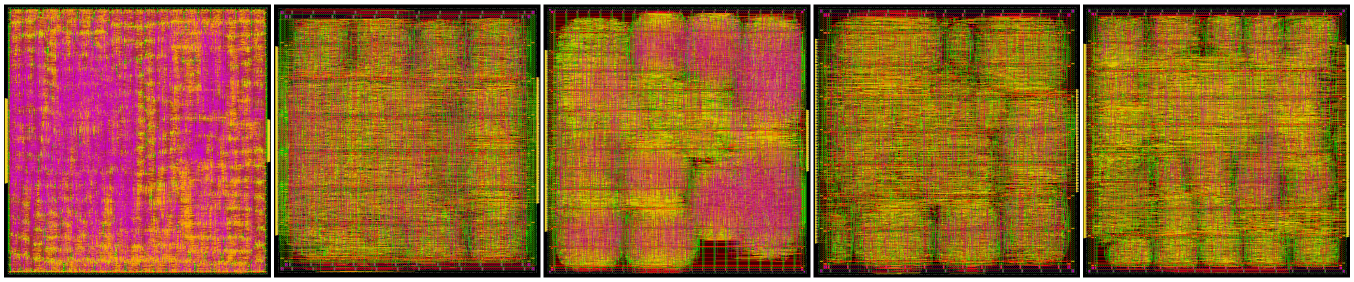


Figure 6: Routed layouts for AES128, Camellia, CAST, MISTY, and SEED (left to right). The same number of metal layers is used for all benchmarks. The highest metal in the stack is M7, represented in pink color.

any location of the design since it is only dependent on the clock and the triggering condition [8]. Furthermore, we consider different versions for each type, as in varying number of triggering bits and number of the payload bits.

An exemplary Trojan with a 16-bit trigger (utilizing the original design) and a 5-bit payload is outlined in Fig. 7. As shown, the Trojan requires only few additional instances which makes it a practical and relevant example; it would likely be hard to spot by conventional Trojan detection. The Trojan's impact on timing is depicted in Fig. 8. The bars in the top subfigure show the number of paths with corresponding timing slack before inserting the Trojan, while the bottom subfigure shows the paths after inserting the Trojan. As shown, the Trojan does not have a considerable impact on the timing, further hindering its detection.

3.4.3 Scoring. Next, we provide more details for the calculation of first-order metrics and scores. The most important settings and considerations are as follows:

(1) Timing checks, logical equivalence, PDN checks, as well as DRC checks, are all hard constraints, i.e., must be met.
  - Thus, as timing checks are based on WNS, only positive slack values are accepted and considered for scoring.

(2) Not considered for scoring are further design checks, like checks for placement and routing issues like dangling wires etc.; see [15] for more details on these checks.
  - Thus, participants can neither improve nor worsen their scores by fixing or worsening those design checks.
  - However, all checks are considered as soft constraints with a margin of +10 issues—any deviation above these margins is considered as a fail.

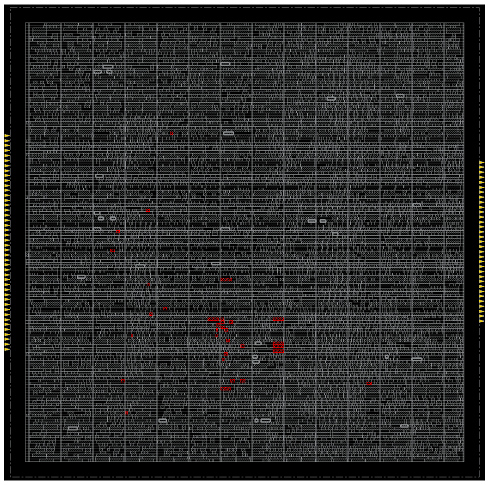


Figure 7: An exemplary Trojan inserted into the SHA256 benchmark. Additional components are highlighted in red.

(3) All metrics are normalized to their respective nominal baseline values, obtained from the provided benchmark layouts. A submission that improves on some metric will be scored a related value between 0 and 1, whereas a deteriorated layout be scored a value greater than 1.
  - For positive WNS values, this means to compute 'baseline_WNS' / 'submission_WNS.'
  - For all other metrics 'm', this means to compute 'submission_m' / 'baseline_m.'

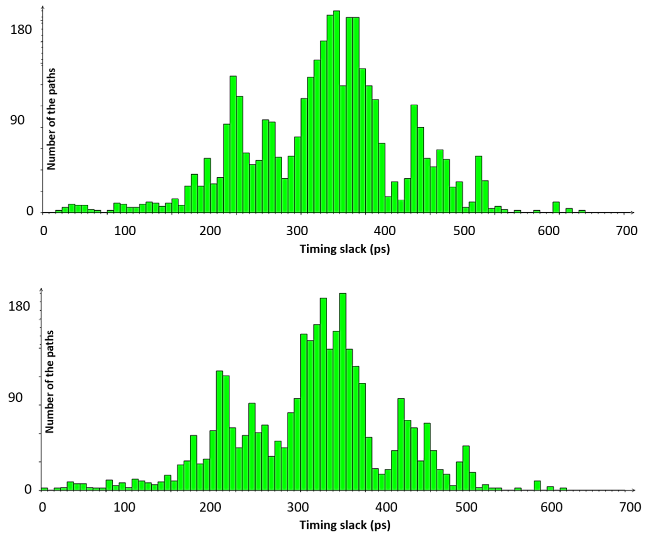


Figure 8: Impact of the inserted Trojan circuitry on timing for the SHA256 benchmark. Distribution of timing paths before Trojan insertion (top) versus after (bottom).

(4) Such normalized scoring is more sensitive to deterioration than it is to improvements. This is on purpose; the main objective is to further improve the layouts, not deteriorate them, so deterioration for any metric(s) should have a relatively large detrimental impact on the overall score.

The actual score calculation is shown below.

$$\begin{aligned} score &= (sec + des)/2 \\ &= ((1/2 \times sec\_ti\_sts + 1/2 \times sec\_ti\_fts) + (des)/2) \end{aligned} \quad (1)$$

with the calculation of score components detailed next, where s refers to the secured/submitted layout and b to the baseline layout.

(1) Trojan insertion – sec_ti
  (a) 50% weighted: placement sites of exploitable regions (sec_ti_sts)
    - 50% weighted: $score(sec\_ti\_sts\_sum) = sec\_ti\_sts\_sum(s)/sec\_ti\_sts\_sum(b)$
    - 33.3% weighted: $score(sec\_ti\_sts\_max) = sec\_ti\_sts\_max(s)/sec\_ti\_sts\_max(b)$
    - 16.6% weighted: $score(sec\_ti\_sts\_med) = sec\_ti\_sts\_med(s)/sec\_ti\_sts\_med(b)$
  (b) 50% weighted: routing resources of exploitable regions (sec_ti_fts)
    - $score(sec\_ti\_fts\_sum) = sec\_ti\_fts\_sum(s)/sec\_ti\_fts\_sum(b)$

(2) Design quality – des
  (a) 33.3% weighted: power (des_pwr)
    - $score(des\_pwr\_tot) = des\_pwr\_tot(s)/des\_pwr\_tot(b)$
  (b) 33.3% weighted: performance (des_prf)
    - 50% weighted: ($des\_prf\_WNS\_set$)
    - 50% weighted: ($des\_prf\_WNS\_hld$)
  (c) 33.3% weighted: area (des_ara)
    - $score(des\_ara\_die) = des\_ara\_die(s)/des\_ara\_die(b)$

3.5 Constraints

For a submission to be considered valid, all the following constraints have to be respected:

- Submissions cannot incorporate trivial defenses. Specifically, filler, decap, and tap cells are scrubbed and thus considered as free placement sites for evaluation of exploitable regions.
- Submissions must meet setup, hold timing checks using the provided SDC files for timing analysis.
- Submissions must have 0 DRC violations.
- Participants must maintain the overall functional equivalence of the underlying design. However, participants are free to revise (parts of) the design implementation, as long as this constraint and the next one are met.
- Participants must maintain the assets, i.e., sensitive components, which are declared along with each benchmark. More specifically, cells declared as assets cannot be removed or restructured. However, participants are free to revise the physical design of assets as well as other logic in general.
- Participants cannot design custom cells; only those cells defined in the provided LIB/LEF files can be utilized.
- Participants cannot revise the metal layers/metal stack.
- Participants must include a clock tree in their submission but are free to revise its implementation, as long as other constraints are met.
- Participants must follow the PDN recipe provided in the reference flow. The PDN structure's stripes are checked for dimensions, area, and locations.
- Submissions must maintain the general IO pin placement. More specifically, pins must remain placed at the left or right side assigned in the baseline layout, but actual pin locations (along the y-axis) can be revised.

4 CONCLUSION

The threat of hardware Trojans has been studied for more than two decades now; yet, this field is still actively researched. On the defensive side, the community lacks commonly adopted approaches for both detecting and preventing Trojans. On the offensive side, there are still doubts about the practicality of Trojans and the real capabilities of such adversaries. This contest, with its focus on advanced security closure against Trojans, is thus an important activity. We seek to educate the physical-design community about (i) hardware security in general, (ii) the key role which CAD tools play toward providing secure and trustworthy ICs, and (iii) the concept of red-team-blue-team security evaluation.

ACKNOWLEDGEMENTS

This work was supported in parts by the Center for Cyber Security (CCS) at NYU New York/Abu Dhabi (NYU/NYUAD) and by the EC through the European Social Fund in the context of the project "ICT programme". The organizers also are thankful for the support of the ISPD Organizing Committee.

REFERENCES

[1] 2007. *Cryptographic Hardware Project.* http://www.aoki.ecei.tohoku.ac.jp/crypto/

[2] 2012. *Freecores: tiny_aes128 implementation.* https://github.com/freecores/tiny_aes

[3] 2013. *Hardware implementation of the SHA-256.* https://github.com/secworks/sha256

[4] 2022. *Reference Design for a modified version of ASAP7nm.* https://github.com/Centre-for-Hardware-Security/asap7_reference_design

[5] Swarup Bhunia, Miron Abramovici, Dakshi Agrawal, Paul Bradley, Michael S. Hsiao, Jim Plusquellic, and Mohammad Tehranipoor. 2013. Protection Against Hardware Trojan Attacks: Towards a Comprehensive Solution. *Des. Test* 30, 3 (2013), 6–17. https://doi.org/10.1109/MDT.2012.2196252

[6] Lawrence T. Clark, Vinay Vashishtha, Lucian Shifren, Aditya Gujja, Saurabh Sinha, Brian Cline, Chandarasekaran Ramamurthy, and Greg Yeric. 2016. ASAP7: A 7-nm finFET predictive process design kit. *Microelectronics Journal* 53 (2016), 105–115. https://doi.org/10.1016/j.mejo.2016.04.006

[7] Vasudev Gohil, Satwik Patnaik, Hao Guo, Dileep Kalathil, and Jeyavijayan (JV) Rajendran. 2022. DETERRENT: Detecting Trojans Using Reinforcement Learning. In *Proc. Des. Autom. Conf.* 697–702. https://doi.org/10.1145/3489517.3530518

[8] Alexander Hepp, Tiago Perez, Samuel Pagliarini, and Georg Sigl. 2022. A Pragmatic Methodology for Blind Hardware Trojan Insertion in Finalized Layouts. In *Proc. Int. Conf. Comp.-Aided Des.* https://doi.org/10.1145/3508352.3549452

[9] W. Hu, C. H. Chang, A. Sengupta, S. Bhunia, R. Kastner, and H. Li. 2020. An Overview of Hardware Security and Trust: Threats, Countermeasures and Design Tools. *Trans. Comp.-Aided Des. Integ. Circ. Sys.* (2020). https://doi.org/10.1109/TCAD.2020.3047976

[10] Susmit Jha and Sumit Kumar Jha. 2008. Randomization Based Probabilistic Approach to Detect Trojan Circuits. In *High Assur. Sys. Eng. Symp.* 117–124. https://doi.org/10.1109/HASE.2008.37

[11] A. B. Kahng and T. Spyrou. 2021. The OpenROAD Project: Unleashing Hardware Innovation. In *Proc. GOMACTech.* https://theopenroadproject.org

[12] Ramesh Karri, Jeyavijayan Rajendran, Kurt Rosenfeld, and Mohammad Tehranipoor. 2010. Trustworthy Hardware: Identifying and Classifying Hardware Trojans. *Computer* 43, 10 (2010), 39–46. https://doi.org/10.1109/MC.2010.299

[13] J. Knechtel. 2021. Hardware Security for and beyond CMOS Technology. In *Proc. Int. Symp. Phys. Des.* https://doi.org/10.1145/3439706.3446902

[14] J. Knechtel. 2022–2023. *Backend Daemon Scripts.* https://github.com/DfX-NYUAD/backend_daemon_gdrive

[15] J. Knechtel et al. 2022–2023. *Advanced Security Closure of Physical Layouts.* https://wp.nyu.edu/ispd23_contest/

[16] J. Knechtel, J. Gopinath, J. Bhandari, M. Ashraf, H. Amrouch, S. Borkar, S.-K. Lim, O. Sinanoglu, and R. Karri. 2021. Security Closure of Physical Layouts. In *Proc. Int. Conf. Comp.-Aided Des.* https://doi.org/10.1109/ICCAD51958.2021.9643543

[17] J. Knechtel, E. B. Kavun, F. Regazzoni, A. Heuser, A. Chattopadhyay, D. Mukhopadhyay, S. Dey, Y. Fei, Y. Belenky, I. Levi, T. Güneysu, P. Schaumont, and I. Polian. 2020. Towards Secure Composition of Integrated Circuits and Electronic Systems: On the Role of EDA. In *Proc. Des. Autom. Test Europe.* https://doi.org/10.23919/DATE48585.2020.9116483

[18] Nimisha Limaye, Nikhil Rangarajan, Satwik Patnaik, Ozgur Sinanoglu, and Kanad Basu. 2022. PolyWorm: Leveraging Polymorphic Behavior to Implant Hardware Trojans. *Trans. Emerg. Top. Comp.* 10, 3 (2022), 1443–1455. https://doi.org/10.1109/TETC.2021.3090060

[19] Eric Love, Yier Jin, and Yiorgos Makris. 2012. Proof-Carrying Hardware Intellectual Property: A Pathway to Trusted Module Acquisition. *Trans. Inf. Forens. Sec.* 7, 1 (2012), 25–40. https://doi.org/10.1109/TIFS.2011.2160627

[20] Satwik Patnaik, Mohammed Ashraf, Ozgur Sinanoglu, and Johann Knechtel. 2019. A Modern Approach to IP Protection and Trojan Prevention: Split Manufacturing for 3D ICs and Obfuscation of Vertical Interconnects. *Trans. Emerg. Top. Comp.* 9 (2019), 1815–1834. Issue 4. https://doi.org/10.1109/TETC.2019.2933572

[21] Tiago Perez and Samuel Pagliarini. 2022. Hardware Trojan Insertion in Finalized Layouts: From Methodology to a Silicon Demonstration. *Trans. Comp.-Aided Des. Integ. Circ. Sys.* (2022). https://doi.org/10.1109/TCAD.2022.3223846

[22] Nikhil Rangarajan, Satwik Patnaik, Johann Knechtel, Shaloo Rakheja, and Ozgur Sinanoglu. 2022. *The Next Era in Hardware Security.* Springer. https://doi.org/10.1007/978-3-030-85792-9

[23] Mohammad Tehranipoor and Farinaz Koushanfar. 2010. A Survey of Hardware Trojan Taxonomy and Detection. *Des. Test* 27, 1 (2010), 10–25. https://doi.org/10.1109/MDT.2010.7

[24] Vinay Vashishtha, Manoj Vangala, and Lawrence T. Clark. 2017. ASAP7 predictive design kit development and cell design technology co-optimization: Invited paper. In *Proc. Int. Conf. Comp.-Aided Des.* 992–998. https://doi.org/10.1109/ICCAD.2017.8203889

[25] Fangzhou Wang, Qijing Wang, Bangqi Fu, Shui Jiang, Xiaopeng Zhang, Lilas Alrahis, Ozgur Sinanoglu, Johann Knechtel, Tsung-Yi Ho, and Evangeline F. Y. Young. 2023. Security Closure of IC Layouts Against Hardware Trojans. In *Proc. Int. Symp. Phys. Des.* https://doi.org/10.1145/3569052.3571878

[26] Kan Xiao and Mohammed Tehranipoor. 2013. BISA: Built-in self-authentication for preventing hardware Trojan insertion. In *Proc. Int. Symp. Hardw.-Orient. Sec. Trust.* 45–50. https://doi.org/10.1109/HST.2013.6581564

[27] K. Yang, M. Hicks, Q. Dong, T. Austin, and D. Sylvester. 2016. A2: Analog Malicious Hardware. In *Proc. Symp. Sec. Priv.* https://doi.org/10.1109/SP.2016.10