Benchmarking Security Closure of Physical Layouts

ISPD 2022 Contest

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

Jayanth Gopinath
jg6476@nyu.edu
New York University
USA

Mohammed Ashraf
ma199@nyu.edu
New York University Abu Dhabi
UAE

Jitendra Bhandari
jb7410@nyu.edu
New York University
USA

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

Ramesh Karri
rkarri@nyu.edu
New York University
USA

ABSTRACT

Computer-aided design (CAD) tools mainly optimize for power, performance, and area (PPA). However, given a large number of serious hardware-security threats that are emerging, future CAD flows must also incorporate techniques for designing secure integrated circuits (ICs). In fact, the stakes are quite high for IC vendors and design companies, as security risks that are not addressed during design time will inevitably be exploited in the field, where vulnerabilities are almost impossible to fix. However, there is currently little to no experience related to designing secure ICs available within the CAD community. For the very first time, this contest seeks to actively engage with the community to close this gap.

The theme of this contest is security closure of physical layouts, that is, hardening the physical layouts at design time against threats that are executed post-design time. More specifically, this contest is focused on selected and seminal threats that, once taken in, are relatively simple to approach and mitigate through means of physical design: Trojan insertion and probing as well as fault injection. Acting as security engineers, contest participants will iteratively and proactively evaluate and fix the vulnerabilities of provided benchmark layouts. Benchmarks and submissions are based on the generic DEF format and related files. Thus, participants are free to use any physical-design tools of their choice, helping us to open up the contest to the community at large.

CCS CONCEPTS

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

KEYWORDS

Physical Design; Contest; Hardware Security; Security Closure; Trojans; Read-Out Attacks; Probing Attacks; Fault-Injection Attacks

ACM Reference Format:
Johann Knechtel, Jayanth Gopinath, Mohammed Ashraf, Jitendra Bhandari, Ozgur Sinanoglu, and Ramesh Karri. 2022. Benchmarking Security Closure of Physical Layouts: ISPD 2022 Contest. In *Proceedings of the 2022 International Symposium on Physical Design (ISPD '22), March 27–30, 2022, Virtual Event, Canada.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3505170.3511046

1 INTRODUCTION

This paper presents the very first contest on *security closure* of physical layouts,[1] i.e., on the challenge of hardening the physical layouts of ICs at design time against various hardware-security threats that are executed post-design time.

This topic is important for multiple reasons. First, many such threats, 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 IC design at higher abstraction layers like logic synthesis, such efforts may be undermined later on by, e.g., 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 [9] 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 electronics is an important but tough endeavour that requires efforts all the way from software applications 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. See also, e.g., [8, 10, 11, 14, 16] for further reading.

[1]There are other hardware-security contests organized by various communities, like the *HACK@EVENT* series [17] or *CSAW* ("see-SAW", the most comprehensive student-run cyber security event in the world, featuring 8 cyber competitions, workshops, and industry events, with final events hosted by 5 global academic centers) [2]. However, none considered so far securing the hardware's "bare metal."

This contest is part of the International Symposium on Physical Design (ISPD) 2022. Participants of this newly introduced theme 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 against different, selected threats. The threats—Trojan insertion and probing, fault injection—represent relatively simple scenarios, with a clear relation to physical design for defending against them. Further, the scope is well limited/constrained for this contest, thereby easing the ramp-up for security-novice participants.

1.2 Objective and General Approach

The objective of this contest is the following. Implement physical-design measures to proactively harden layouts against:

(1) post-design insertion (i.e., during mask generation or manufacturing) of Trojans that are implemented at the gate level;
(2) in-field electro-optical and contact-based probing as well as fault injection attacks, all targeting at the IC's frontside (i.e., the metal layers).

See Sec. 2.1 for context and more details on these threat scenarios.

To achieve this objective, participants would want to, e.g., revise placement and routing in such a way that insertion of Trojan components as well as probing, fault injection on particular devices or wires becomes difficult, all while also accounting for the impact on design quality induced by the defense measures. 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 solutions. See also Sec. 3.2.1 for some trivial example solutions.

1.3 Logistics

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

The benchmarks and submissions are based on DEF and related files; see Sec. 3.2 and the contest website [9] for more details. Thus, participants can work on any physical-design platform of their choice, be it commercial tools, open-source tools like *OpenROAD* [7], or custom in-house tools. Before integrating some defense schemes into their platform, participants would want to i) fully understand the scope in general and the threats in particular (Sec. 2), ii) fully understand the way the threats are considered and scored for this contest (Sec. 3.3), and iii) be as creative as possible while not "re-inventing the wheel" for core algorithms and design techniques.

There is an alpha round, where we provide a public set benchmarks, with results and rankings published regularly and feedback provided to the participants, to spur the contest. All participants that submit some valid solution for each benchmark move on the final round. There, we provide further sets of benchmarks as mix of public as well as blind benchmarks, covering a variety of designs and layout complexities. The final results, rankings, and awards will be first announced at ISPD and then published on the contest website [9] as well. 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.

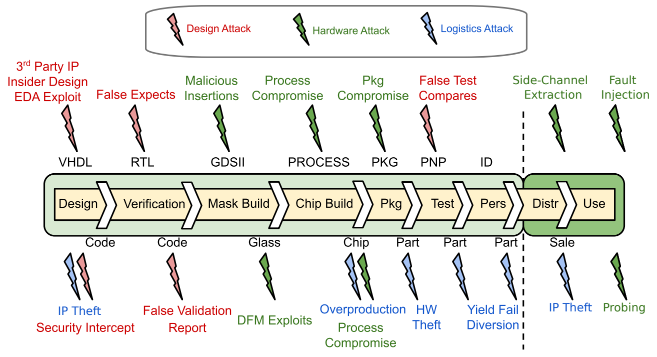

Figure 1: The IC supply-chain and life-cycle with various threats affecting different stages stages. Adopted from [3].

2 BACKGROUND AND TASKS DESCRIPTION

2.1 Hardware Security

There are various challenges or rather threats to consider when we talk about hardware security. An overview on these threats, linked to the different stages of the supply-chain and life-cycle of ICs, is shown in Fig. 1.

The main threats of interest to the physical-design community are i) Trojans, ii) side-channel attacks, iii) fault-injection attacks, iv) probing attacks, and v) IP piracy [8, 16]. Typically, each kind of threat is further divided/categorized; for example, there is direct physical fault injection, e.g., using laser light [18], voltage glitches, etc., versus indirect fault injection, e.g., repetitive writing to physical memory locations (also known as "Rowhammer" attack [15]). Some threats share a similar physical attack vector, like laser fault injection and laser-assisted optical probing [13].

2.1.1 Hardware Trojans. Trojans are malicious hardware modifications [16, 20]. The notion of Trojans is diverse, covering malicious modifications that are: i) targeting at the system level, gate level, interconnects level, transistor level, and/or the physical level; ii) seeking to leak information from an IC, reduce the IC's performance, or disrupt an IC's working altogether; iii) are always on, triggered internally, or triggered externally. Most Trojans comprise a trigger and a payload; the trigger activates the payload on attack conditions, and the payload serves to perform the actual attack. Since IC supply-chains are largely outsourced nowadays, adversaries at various entities could introduce such Trojans, e.g., through untrustworthy third-party IP, by adversarial designers, during mask generation or manufacturing, or even during deployment of ICs.

As indicated, for this contest, we focus on post-design insertion (i.e., during mask generation or manufacturing) of Trojans that are implemented at the gate level. (Thus, we exclude advanced Trojans that are, e.g., implemented at the interconnects level [14].) An example of such relatively simple Trojan is shown in Fig. 2.

The related task for this contest is to proactively harden the layouts against such Trojan insertion. This means to, e.g., control placement and routing in such a way that insertion of Trojan components becomes difficult, but also considering impact on design

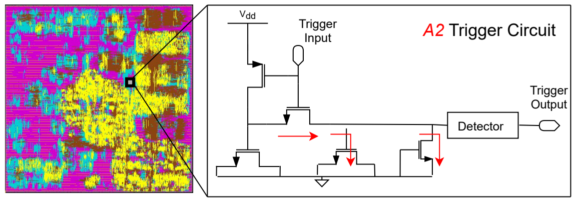

Figure 2: Exemplary IC layout of an *OR1200* processor design with the *A2* Trojan embedded [21]. The zoom-in highlights the additional logic inserted for the Trojan trigger. The payload of the *A2* Trojan, maliciously setting the privilege mode of the *OR1200* processor, is not shown here.

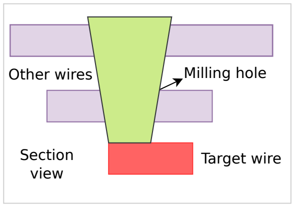

Figure 3: Simplified working principle of frontside attack. After milling through wires that are obstructing direct access to some target wires, the attacker can use, e.g., contact-based micro-probing needles on the target wires.

quality of such measures at the same time. To enable a fair contest, we have to restrict the scope of defense efforts to the physical design stages. Thus, we do not allow to, e.g., introduce dedicated sensor circuitry. See also Sec. 3 for more details on constraints as well as for some guidance for permissible defense efforts.

2.1.2 Probing, Fault Injection. Probing attacks extract data from devices or wires by probing through the metal frontside or the substrate backside [5, 13, 16]. Such attacks are enabled by different means, mainly contact-based micro-probing, electro-magnetic field probing, or electro-optical device probing. These means leverage various physical vectors, i.e., electro-magnetic field emissions, electrical charges, photon injection and emission, etc. Probing attacks have their roots in failure analysis techniques, hence also apply for advanced nodes, though requiring more efforts there. Some attacks like micro-probing require line of sight and direct access to the device/wire of interest; thus, such attacks are often complemented by techniques like focused ion beam milling.

The working principle for an exemplary probing attack is shown in Fig. 3. Note the conical shape of the milling intrusion. Also note the potential challenge for an attacker, while reaching to the target wire, to avoid cutting other critical wires, e.g., related to attack detection in particular or to stable IC operation in general.

Fault-injection attacks induce faults to aid deducing sensitive information [15, 16, 18]. Therefore, fault injection can support side-channels attacks or other analytical attacks. Fault-injection attacks cover direct fault injection, e.g., using laser light or electromagnetic waves, as well as indirect fault injection, e.g., by repetitive writing to particular memory locations or by deliberate misuse of dynamic voltage and frequency scaling (DVFS) features.

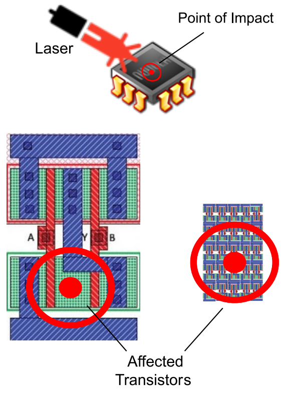

Figure 4: Laser fault injection on devices, adopted from [15]. Left: standard cells (250nm; height 12.5μm) hit by a 5μm laser spot. Right: standard cells (28nm; height 1.2μm) hit by the same laser. Note that any higher metal layers, blocking the laser fault injection, are not shown here.

The working principle for an exemplary fault-injection attack is shown in Fig. 4. Here, laser light is injected into the active layers of two ICs implemented using different technology nodes, hence the scope of the fault injection differs: for older nodes, individual transistors can be targeted at, whereas for newer nodes, multiple adjacent cells will be targeted at once, which may complicate orchestrated attack schemes. It is important to note that this aspect is *not* specifically handled in this contest; we do not consider varying technology nodes and/or laser tools.

As indicated, for this contest, we focus on probing and fault injection attacks targeting at the frontside. We understand that attacks targeting at the backside are more capable and practical at the same time [5, 13]. However, defending the backside requires dedicated circuitry or technology support (e.g., using current sensors or 3D integration [12]), whereas defending the frontside can be realized via physical-design efforts. Note that electro-optical, contact-based probing and fault injection targeting at the frontside share the same attack principle, namely to “sneak through” the metal layers down to active devices or wires of interest.

The related task for this contest is to proactively harden the layouts against in-field probing and fault injection attacks that are targeting the frontside. This means, e.g., to revise placement and routing such that access to sensitive devices or wires becomes difficult, namely by nature of (other) metal segments obstructing the line of sight required for such attacks. At the same time, the impact on design quality of such defense measures must be considered. Again, to enable a fair contest, we have to restrict the scope to physical-design stages. Thus, we do not allow to, e.g., introduce dedicated sensor circuitry. See also Sec. 3 for more details on constraints as well as for some guidance for permissible defense efforts.

2.2 Security Closure

This contest is focused on security closure of physical layouts, that is, on hardening the physical layouts at design time against various threats that are executed post-design time. As discussed, this topic is important for multiple reasons. First, many threats,

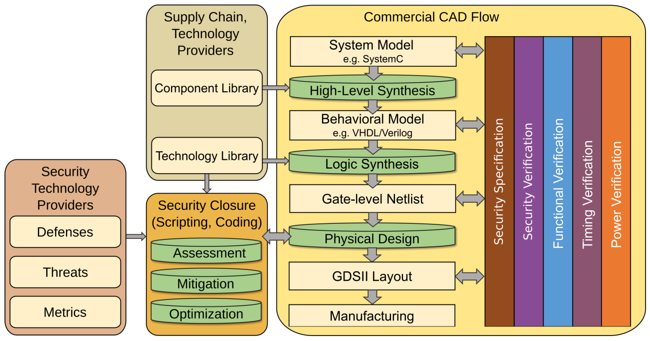

Figure 5: Secure-by-design CAD flow with integrated means for security closure, security specification, security verification, and security providers. Adopted from [10].

like Trojan insertion or side-channel attacks, are directly targeting 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 IC design at higher abstraction layers, like high-level synthesis or logic synthesis, such efforts may be undermined by, e.g., PPA optimization, thus becoming futile without dedicated support for security closure at layout level.

Secure-by-design and *security closure* are two related, emerging paradigms for CAD tools [10, 11, 14]. The secure-by-design paradigm means to support i) top-down propagation and translation of security requirements and specifications, as well as ii) bottom-up verification and validation of defenses against attacker's capabilities and limitations (the latter obtained from security technology providers). Security closure is the specific paradigm for the physical-design stages, loosely/conceptionally similar to other sign-off stages like timing closure but focused on security. Means for security closure will be based on ECO placement, routing, etc., as needed. A secure-by-design CAD flow is outlined in Fig. 5.

3 IMPLEMENTATION AND EVALUATION

3.1 Platform

3.1.1 Tool Flow for Participants. As indicated, efforts for this contest are to be focused on physical design. The participants' envisioned measures for security closure against the different threats considered in this contest should ideally be streamlined; any IC layout has to be hardened against different threats at once. At the same time, the impact on design quality must be considered as well. Given that there are various, quite different metrics to be considered for design quality and security closure at once (Sec. 3.3), some machine learning-based guidance can be promising here.

Recall that participants are free to use any physical-design tools of their choice, be they commercial, open-source, or own in-house tools. As such, we aim to open up this contest to a broader audience. At the same time, we understand that, for participants without any such tool flow available, the ramp-up for this contest would be considerable. In such cases, we suggest to focus on established flows like *OpenROAD* [7].

3.1.2 Backend for Organizers. Our implementation and evaluation backend is based on commercial tools, in particular *Cadence Innovus*. The metrics, scoring, and file management tasks for contest preparation as well as contest evaluation are all implemented via scripting, using *tcl* and *bash* in particular.

Our backend is not made public for this contest, for the following reasons. First, we do not want participants to require access to and experience with our tool setting, especially with *Cadence Innovus*, let alone our custom scripting. Second, the backend itself is part of an ongoing research project where code releases are not approved yet. In any case, we provide the participants with extensive details and Q&A interaction on metrics, scoring, benchmarks, and implementation constraints. Thus, the lack of access to the actual backend should not constitute any disadvantage; it might even rather ease the burden, by allowing participants to directly focus on their ideas and implementation.

3.1.3 Frontend for Participants. Given the above outlined setting, 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, private Google Drive folder. Teams may upload submission files anytime, upon which new files are automatically downloaded to our backend servers for evaluation. Results will be returned into the teams' respective folders. Results will include the overall score but also report files as generated by our backend, to provide participants with more detailed insights. Participants will also be send email notifications once new evaluation results are available, as processing may take some time once the backend servers are put under load from multiple parallel submissions.

3.2 Benchmarks

Recall that this contest is the very first which is focused on security closure of physical layouts in particular (and even on layout-level security challenges in general). Thus, we had to devise our own set of benchmarks (i.e., physical layouts) as follows.

(1) We select relevant designs as HDL code. We obtain:
   (a) crypto cores (i.e., *Camellia*, *CAST*, *MISTY*, *SEED*, *TDEA*) from [19];
   (b) the *openMSP430* microcontroller from [4];
   (c) the *MIT-LL CEP SoC*, i.e., a security-centric SoC design with crypto, DSP, GPS, and key management modules, among others, from [6]; and
   (d) further crypto cores (i.e., *PRESENT*, *SPARX*) from in-house projects.
(2) All designs are synthesized and implemented. Mainly to ease the ramp-up for participants, we use the well-known and publicly available *Nangate 45nm Open Cell Library* [1]. We use *Synopsys DC* for synthesis and *Cadence Innovus* for layout implementation. We use different timing constraints, target utilizations, and library configurations (process corners, metal stacks) across the different designs, to make for a varied range of benchmark layouts.
(3) We identify security assets from the post-layout netlists, based on the documentation and HDL code of the designs.

For example, we identify cells and nets which belong to key-memory registers or key-control logic as such assets. Note that our selection of assets is not exhaustive. In fact, our selection varies on purpose across different benchmarks, to provide for a range of simple to challenging benchmarks for security closure.

In short, the benchmarks are physical layouts of different security-centric designs that have varying ranges of complexity, size (in terms of cells, nets), utilization, timing constraints, number of assets, considered corners, and available metal layers.

All benchmarks are bundled as archive files, containing the DEF file and other supportive files: the post-layout netlist directly corresponding to the DEF, the SDC and MMMC files used for timing analysis, the custom list files for sensitive cells and nets, the customized LIB/LEF files, various snapshots with sensitive cells and nets highlighted, and a README). Furthermore, we include the reports obtained by our evaluation. Specifically, aside from regular PPA reports, custom-generated reports provide insights on i) the areas of sensitive cells and wires that are exposed to probing and fault injection, and ii) placement sites and routing resources of regions that are exploitable for Trojan insertion. More details on these evaluation metrics are provided in Sec. 3.3.

Over the course of the contest, we release three different sets of benchmarks: sample benchmarks, alpha-round benchmarks, and final-round benchmarks, as described next. All benchmarks are released on the contest website [9] and will remain online there after the contest concludes. With the contest results to be published online there as well, we hope to further stimulate competitive interest and efforts from the community after the contest concludes.

3.2.1 Sample Benchmarks. This set contains three different layout implementations of the same *AES* design (module extracted from the *MIT-LL CEP SoC*). This set is meant as "warm-up" and introduction to the contest theme. Thus, the three layouts represent baseline versus somewhat hardened implementations. More specifically, we provide: i) a baseline layout with 70% utilization, ii) a baseline layout with 90% utilization, and iii) a layout with 70% utilization and some shielding of cells and nets against probing, fault injection attacks.

The relevant insights for participants would be the following:

- A naive increase of utilization from 70% to 90% helps to harden the layout against Trojan insertion (Fig. 6), but makes timing closure much more difficult (not illustrated here).
- Shielding of sensitive cells and nets helps to harden the layout against probing and fault injection, but makes timing closure more difficult as well, along with further PPA cost.
- Competitive schemes should achieve both, hardening layouts and limiting impact on design quality. Such efforts are explicitly not made for the sample benchmarks.

3.2.2 Alpha-Round Benchmarks. This set is made public at the beginning of the alpha round. This set contains layouts of various crypto cores (namely *CAST*, *Camellia*, *MISTY*, *PRESENT*, and *AES*). Note that the *AES* layouts from the sample benchmarks are also included here, but with a different sets of cell and net assets to be protected. As indicated, the layouts have varying ranges of complexity, size, utilization, timing constraints, number of assets, and available metal layers. Figure 7 illustrates the different scales of cell and net assets for two exemplary crypto cores.

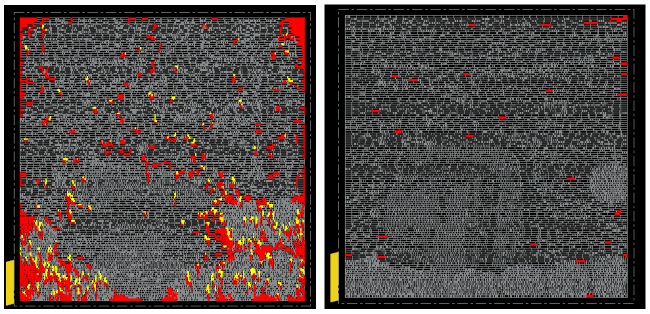

Figure 6: Exploitable regions for the *AES* layouts with 70% utilization (left) versus 90% utilization (right). Regions with 50%+ free routing tracks are highlighted in red, whereas regions with less free tracks are highlighted in yellow.

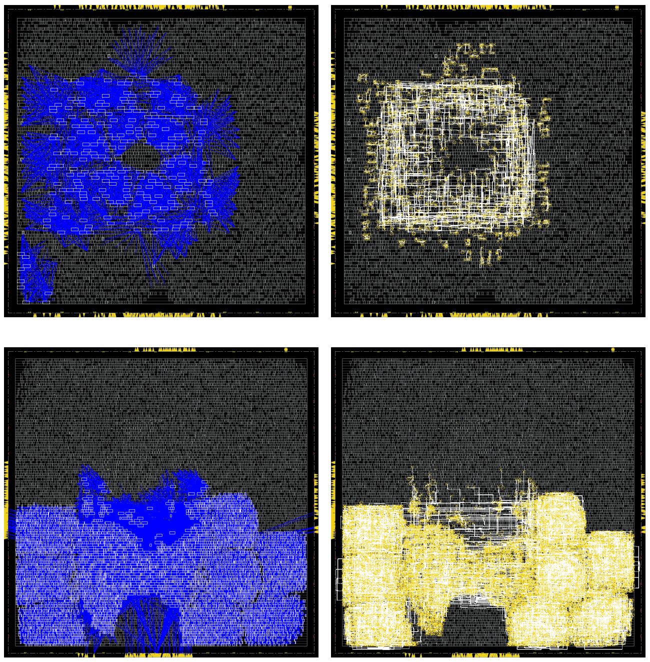

Figure 7: Cells assets (left) and nets assets (right) highlighted for the baseline layouts of exemplary crypto cores *Camellia* (top) versus *CAST* (bottom). The *Camellia* benchmark represents a relatively simple case, with 265/6,710 cells and 384/7,074 nets labelled as assets, whereas the *CAST* benchmark represents a larger and more challenging case, with 4,805/12,682 cells and 4,935/13,050 nets labelled as assets.

3.2.3 Final-Round Benchmarks. This set is split into two; one is made public at the beginning of the final round, the other remains undisclosed until the very end, such that participants cannot tweak their efforts for particular benchmark characteristics (if any).

The public part contains the *openMSP430* microprocessor and various crypto cores (namely those from the alpha round but now with more aggressive utilization and timing constraints). The part undisclosed-til-end contains the large-scale *CEP SoC* and the remaining crypto cores (namely *SEED*, *TDEA*, and *SPARX*). The *CEP SoC* is provided in two versions, namely one as loose and one as aggressive implementation. The crypto cores are provided as medium-to-aggressive implementations. Thus, as before, the layouts vary in complexity, size, utilization, timing constraints, number of assets, and available metal layers.

3.3 Metrics and Scoring

The scoring considers multiple security and design metrics in a streamlined manner, as explained here in detail. Besides, there are various constraints to be followed, as outlined in Sec. 3.4.

3.3.1 Trojan Insertion. Related metrics are based on so-called *exploitable regions*, i.e., sets of continuous placement sites that are exploitable because they are i) free, ii) occupied only by filler or other non-functional cells, or iii) occupied by functional but unconnected cells. Free routing tracks are also considered, as Trojans would require such as well. In other words, exploitable regions are where an attacker would be able to find or make some space, along with routing resources, to place and wire up their Trojans.

For this contest, without loss of generality, exploitable regions are defined as sets of 20 or more exploitable placement sites extending in continuous manner in horizontal and/or vertical direction into any polygonal shape. Such assumption is practical as an attacker would need at least a few sites for Trojan insertion—the specific number of 20 sites is taken from the *A2* Trojan [21]—but it is also simplified for the sake of generic evaluation (i.e., without having to consider actual Trojan implementations). Note the following (non-exhaustive) implications of this simplified assumption.

(1) Depending on the layout, these 20+ sites may be arranged in largely vertical shape, i.e., across rows, which would be more difficult to exploit for Trojan placement. However, such regions are potentially still relevant, e.g., for Trojan routing assuming that exploitable sites would hold more free routing tracks on average than occupied sites.
(2) The selection of continuous sites is strictly related to any layout as is; it does not account for the fact that an attacker might be able to rearrange some nearby cells to further extend the number of continuous sites. However, such efforts by an attacker are still subject to timing constraints; the latter implication is further discussed next.

Exploitable regions are defined and evaluated only within an *exploitable distance* of cells representing security assets. That is because Trojans need to be placed and routed such that timing requirements of the original design are still met; thus, placement sites which are closer to cell assets tend to be more vulnerable.

More specifically, the exploitable distances are determined as follows. For paths related to the cell asset, the positive timing slack (if any) is determined. Next, the delay impact of adding an additional NAND gate—representing an exemplary, most simple form of Trojan—is determined, by analyzing wiring delays and the library for different output loads and input transition times. To simplify and limit runtime for evaluation, we do not perform actual routing here; we estimate wiring loads based on Manhattan distance. Then we derive what maximal distance the hypothetical NAND-gate Trojan's routing could afford while consuming just the positive slack available across the paths related to the cell asset, but not more (i.e., still meet timing).

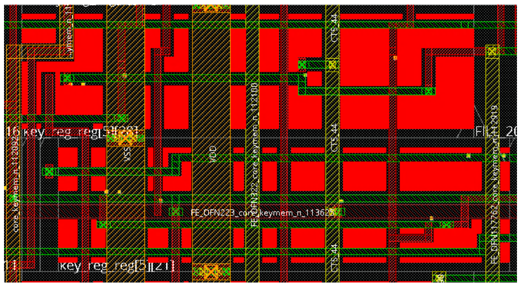

Figure 8: Example for standard cells' exposed area, highlighted in red, zoomed in. Routing across the different metal layers blocks the exposed areas somewhat, but there are still large gaps; the cells remain vulnerable to large degrees.

3.3.2 Frontside Probing, Fault Injection. Related metrics are based on the so-called *exposed area* of cells and nets, i.e., any spatial region, be it continuous or fragmented, of those cell/net assets which is accessible via direct line of sight through the metal stack. An example for some standard cells is provided in Fig. 8.

The notion of exposed area is appropriate for a generic evaluation against frontside probing, fault injection attacks for the following (non-exhaustive) list of reasons.

(1) Exposed area is conservative; it assesses any first-order vulnerability based on direct line of sight to the asset.
(2) Exposed area is agnostic to the attacker's capabilities, since regions of any size and fragmentation are accounted for. This is a practical simplification as follows. For example for laser fault injection on cells, it is sufficient to have small transistor-sized regions exposed to induce some faults, as the maliciously acting photocurrents induced by the laser can be well controlled through the attacker's setup. At the same time, with more regions exposed, the attack surface literally increases, and the options for targeted fault injections increase. Similar reasoning applies to other attacks and net/wire assets. Thus, in general, the larger the exposed area, the more vulnerable the assets.

3.3.3 Design Quality. Metrics are focused on standard metrics, describing power, performance, and area, as well as routing.

3.3.4 Metrics Classification. A short description of all considered metrics, along with their variable names listed in italic, is given in the following classification.

(1) Security – *sec*
  (a) Trojan insertion – *ti*
    (i) Placement sites of exploitable regions (ers)
      - Max # of sites across all ers – *sts_max*
      - Avg # of sites across all ers – *sts_avg*
      - Total # of sites across all ers – *sts_total*

(ii) Routing resources of exploitable regions (ers)
  - Max # of free tracks across all ers – fts_max
  - Avg # of free tracks across all ers – fts_avg
  - Total # of free tracks across all ers – fts_total
  - Note that, for each exploitable region, free tracks are summed up across all metal layers.[2]

(b) Frontside probing and fault injection – fsp_fi

(i) Exposed area (ea) of standard cell assets
  - Max % of ea across all cell assets – ea_c_max
  - Avg % of ea across all cell assets – ea_c_avg

(ii) Exposed area (ea) of net assets
  - Max % of ea across all net assets – ea_n_max
  - Avg % of ea across all net assets – ea_n_avg
  - Total ea across all net assets – ea_n_total

(2) Design quality – des

(a) Power
  - Total power – p_total

(b) Performance
  - Worst neg. slack, setup timing req. – $setup\_WNS$
  - Total neg. slack, setup timing req. – $setup\_TNS$
  - # of failing endpoints, setup timing req. – $setup\_FEP$

(c) Area
  - Total die area (*not* standard cell area) – die_area

(d) Routing
  - # of DRC violations – DRC

3.3.5 Scoring. The score calculation is given below. First, however, see the following important considerations for scoring:

(1) All metrics are grouped following the above classification. All metrics are weighted and summed up within their related group. Weighting is also applied across the upper levels in the classification hierarchy.

(2) For the overall score, we consider the product of (summed and weighted) metrics for Trojan insertion and frontside probing, fault injection as well as (summed and weighted) metrics for design quality. Thus, security and design quality are to be optimized at once, or rather security should be incorporated without negative impact on design quality.

(3) All metrics are normalized to their respective nominal baseline, i.e., obtained from the provided benchmark layouts.
  - Some metrics require case-specific definitions for normalized scoring. Details are given with the score calculation.
  - A layout that improves on some metric is scored a related value between 0 (max improvement) and 1 (min improvement), whereas a deteriorated layout will be scored >1.[3]
  - Such normalized scoring is more sensitive to deterioration than it is to improvements. This is on purpose—we want to further improve the layouts, not deteriorate them. Note that it is *not* important whether some benchmark layouts are already aggressively optimized in terms of design quality or not; a more (versus less) aggressive benchmark layout will just provide less (versus more) flexibility for optimizing both on security and design quality.

The overall score, to be minimized, is defined as:

$$score = sec \times des = ti \times fsp\_fi \times des \qquad (1)$$

with the calculation of score components detailed next. In the following, note that s refers to the secured/submitted layout and b refers to the corresponding baseline layout.

(1) Trojan insertion – ti

(a) 60% weighted: placement sites of exploitable regions
  - 50% weighted: $score(sts\_total) = sts\_total(s)/sts\_total(b)$
  - 33.3% weighted: $score(sts\_max) = sts\_max(s)/sts\_max(b)$
  - 16.6% weighted: $score(sts\_avg) = sts\_avg(s)/sts\_avg(b)$

(b) 40% weighted: routing resources of exploitable regions
  - 50% weighted: $score(fts\_total) = fts\_total(s)/fts\_total(b)$
  - 33.3% weighted: $score(fts\_max) = fts\_max(s)/fts\_max(b)$
  - 16.6% weighted: $score(fts\_avg) = fts\_avg(s)/fts\_avg(b)$

(2) Frontside probing and fault injection – fsp_fi

(a) 50% weighted: exposed area of standard cell assets
  - 60% weighted: $score(ea\_c\_max) = ea\_c\_max(s)/ea\_c\_max(b)$
  - 40% weighted: $score(ea\_c\_avg) = ea\_c\_avg(s)/ea\_c\_avg(b)$

(b) 50% weighted: exposed area of net assets
  - 50% weighted: $score(ea\_n\_total) = ea\_n\_total(s)/ea\_n\_total(b)$
  - 33.3% weighted: $score(ea\_n\_max) = ea\_n\_max(s)/ea\_n\_max(b)$
  - 16.6% weighted: $score(ea\_n\_avg) = ea\_n\_avg(s)/ea\_n\_avg(b)$

(3) Design quality – des

(a) 10% weighted: power
  - $score(p\_total) = p\_total(s)/p\_total(b)$

(b) 30% weighted: performance
  - 50% weighted: $setup\_TNS$ (notation is consolidated below into $setup\_NS$)
  - 33.3% weighted: $setup\_WNS$ (notation is consolidated here into $setup\_NS$)
    - If $setup\_NS(s) < 0, setup\_NS(b) < 0$: $score(setup\_NS) = setup\_NS(s)/setup\_NS(b)$
    - Else if $setup\_NS(s) < 0, setup\_NS(b) >= 0$: $score(setup\_NS) = min(setup\_NS\_max\_cost, abs(setup\_NS(s)))$ with $setup\_NS\_max\_cost = 3$ as default
    - Else (i.e., $setup\_NS(s) >= 0$): $score(setup\_NS) = 0$

[2]While metal1 is particularly relevant for an attacker to connect to the pins of their Trojan cells, free tracks in other layers would also be relevant, namely to tap into/connect with some sensitive nets and to complete intra-Trojan routing. Hence, for any exploitable region, we cannot say in advance which layers are most relevant (i.e., should be weighted more) until we evaluate all the sensitive nets' routing tracks and even more so until we study the routability of various possible Trojans being inserted in such region—these are impractical tasks. Thus, we have accordingly simplified the fts_* metrics to account for the sum of free tracks across all metal layers.

[3]This does not always hold true. Specifically, in case *both* baseline and submission metrics have a value of 0, where no improvement is achieved, related scores are 0 by definition (and not 1). This is needed to avoid particular noncontinuous scaling behaviour that could otherwise be exploited for cheating.

- 16.6% weighted: $setup\_FEP$
  - If $setup\_FEP(b) > 0$:
    $score(setup\_FEP) = setup\_FEP(s)/setup\_FEP(b)$
  - Else (i.e., $setup\_FEP(b) = 0$):
    $score(setup\_FEP) =$
    $min(setup\_FEP\_max\_cost, setup\_FEP(s))$
    with $setup\_FEP\_max\_cost = 3$ as default

(c) 30% weighted: area
- $score(die\_area) = die\_area(s)/die\_area(b)$

(d) 30% weighted: routing
- If $DRC(b) > 0$:
  $score(DRC) = DRC(s)/DRC(b)$
- Else (i.e., $DRC(b) = 0$):
  $score(DRC) = min(DRC\_max\_cost, DRC(s))$
  with $DRC\_max\_cost = 3$ as default

Note that the weighting of security metrics emphasizes total over max over avg values, respectively; this is to guide toward minimizing the overall and worst-case vulnerabilities. For design quality, note that performance, area, and routing are considered equally relevant, whereas power is considered less relevant. This aims to honor the sensitivity of performance, area, and routing for design closure in general, while providing some "slack" toward security-closure efforts. In this context, also note that we do not account for any loss or gain of positive timing slack; thus, we implicitly release such slack (if any) for security-closure efforts.

3.4 Constraints

For a fair and focused competition, participants have to adhere to some constraints as follows. Note that compliance is automatically checked within our evaluation backend. Violations are reported to the participants, and repeated cases of excessive violations may lead to disqualification.

- Participants must maintain the functional equivalence of the design. However, participants are free to revise parts of the design, as long as this and the next constraint are met.
- Participants must maintain the sensitive components, which are declared along with each benchmark. More specifically, cells and nets declared as assets cannot be restructured.
- Participants cannot add dedicated, custom circuitry (e.g., sensors to detect laser fault injection).
- Participants cannot design custom cells; only those cells defined in the provided LIB/LEF files can be utilized.
- Participants cannot add additional metal layers (e.g., to trivially protect against frontside probing attacks).
- Participants cannot move the PG network to different layers. Furthermore, participants need to maintain i) the overall structure of the PG network as well as ii) the ratio of routing tracks occupied by the PG network to total routing tracks. Deviations of up to +/- 5% in the ratio of track utilization by the PG network is permissible.
- Participants must maintain the relative IO pin placement. Note that re-scaling of the die outline is allowed; if done, however, the IO pins must afterwards be placed in similar relative coordinates. Deviations of up to +/- 5% along the horizontal/vertical boundary are permissible.
- Participants cannot incorporate trivial defenses; specifically, filler cells, other non-functional cells, and functional but unconnected cells are considered as free placement sites for evaluation of exploitable regions.

ACKNOWLEDGEMENTS

This work was supported in parts by the Center for Cyber Security (CCS) at NYU New York/Abu Dhabi (NYU/NYUAD). We like to thank the ISPD committee for featuring this contest, and David Chinnery, Gracieli Posser in particular for interaction and guidance. We also like to thank Sumesh Manjunath Ramesh (NYUAD) for sharing HDL codes for the *PRESENT* and *SPARX* crypto cores.

REFERENCES

[1] Nangate Inc 2011. *NanGate FreePDK45 Open Cell Library.* Nangate Inc. http://www.nangate.com/?page_id=2325
[2] NYU Center for Cybersecurity 2021. *CSAW: Cyber Security Awareness Week.* NYU Center for Cybersecurity. https://www.csaw.io
[3] K. Bernstein. 2016. *Rapid Authentication Through Verification, Validation, and Marking.* Technical Report. DARPA. https://www.ndia.org/-/media/sites/ndia/meetings-and-events/divisions/trusted-micro-electronics--joint-working-group/687c-aug-2016/2-bernstein---distro-a.ashx
[4] O. Girard et al. 2021. *openMSP430 at opencores.org.* https://opencores.org/projects/openmsp430
[5] C. Helfmeier, D. Nedospasov, C. Tarnovsky, J. S. Krissler, C. Boit, and J.-P. Seifert. 2013. Breaking and Entering Through the Silicon. In *Proc. Comp. Comm. Sec.* https://doi.org/10.1145/2508859.2516717
[6] M. Hicks, P. Fiscarelli, B. Chetwynd, et al. 2021. *MIT-LL Common Evaluation Platform (CEP) at github.com.* https://github.com/mit-ll/CEP
[7] A. B. Kahng and T. Spyrou. 2021. The OpenROAD Project: Unleashing Hardware Innovation. In *Proc. GOMACTech.* https://theopenroadproject.org
[8] J. Knechtel. 2021. Hardware Security for and beyond CMOS Technology. In *Proc. Int. Symp. Phys. Des.* https://doi.org/10.1145/3439706.3446902
[9] J. Knechtel et al. 2022. *Security Closure of Physical Layouts.* https://wp.nyu.edu/ispd_22_contest
[10] 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
[11] 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
[12] J. Knechtel, S. Patnaik, and O. Sinanoglu. 2019. 3D Integration: Another Dimension Toward Hardware Security. In *Proc. Int. On-Line Test Symp.* https://doi.org/10.1109/IOLTS.2019.8854395
[13] T. Krachenfels, H. Lohrke, J.-P. Seifert, E. Dietz, S. Frohmann, and H.-W. Hübers. 2020. Evaluation of Low-Cost Thermal Laser Stimulation for Data Extraction and Key Readout. *J. Hardw. Sys. Sec.* 4, 1 (2020). https://doi.org/10.1007/s41635-019-00083-9
[14] J. Lienig, S. Rothe, M. Thiele, N. Rangarajan, M. Ashraf, M. Nabeel, H. Amrouch, O. Sinanoglu, and J. Knechtel. 2021. Toward Security Closure in the Face of Reliability Effects. In *Proc. Int. Conf. Comp.-Aided Des.* https://doi.org/10.1109/ICCAD51958.2021.9643447
[15] O. Mutlu and Jeremie S. Kim. 2020. RowHammer: A Retrospective. *Trans. Comp.-Aided Des. Integ. Circ. Sys.* 39, 8 (2020). https://doi.org/10.1109/TCAD.2019.2915318
[16] N. Rangarajan, S. Patnaik, J. Knechtel, S. Rakheja, and O. Sinanoglu. 2021. *The Next Era in Hardware Security.* Springer. https://doi.org/10.1007/978-3-030-85792-9
[17] A.-R. Sadeghi et al. 2021. *Hardware Security CTF Competitions.* https://hackatevent.org
[18] B. Selmke, J. Heyszl, and G. Sigl. 2016. Attack on a DFA Protected AES by Simultaneous Laser Fault Injections. In *Proc. Worksh. Fault Diag. Tol. Cryptogr.* https://doi.org/10.1109/FDTC.2016.16
[19] T. Sugawara, N. Homma, T. Aoki, and A. Satoh. 2007. ASIC Performance Comparison for the ISO Standard Block Ciphers. In *Proc. JWIS.* http://www.aoki.ecei.tohoku.ac.jp/crypto/web/cores.html
[20] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor. 2016. Hardware Trojans: Lessons Learned After One Decade of Research. *Trans. Des. Autom. Elec. Sys.* 22, 1 (2016). https://doi.org/10.1145/2906147
[21] 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