Games, Dollars, Splits: A Game-Theoretic Analysis of Split Manufacturing

Vasudev Gohil, *Graduate Student Member, IEEE,* Mark Tressler, Kevin Sipple, Satwik Patnaik, *Member, IEEE,* and Jeyavijayan (JV) Rajendran, *Senior Member, IEEE*

***Abstract*—Split manufacturing has been proposed as a defense to prevent threats like intellectual property (IP) piracy and illegal overproduction of integrated circuits (ICs). Over the last few years, researchers have developed a plethora of attack and defense techniques, creating a cat-and-mouse game between defending designers and attacking foundries. In this paper, we take an orthogonal approach to this ongoing research in split manufacturing; rather than developing an attack or a defense technique, we propose a means to analyze different attack and defense techniques. To that end, we develop a game-theoretic framework that helps researchers evaluate their new and existing attack and defense techniques. We model two attack scenarios using two different types of games and obtain the optimal defense strategies. We perform extensive simulations with our proposed framework, using nine different attacks and a class of placement and routing-based defense techniques on various benchmarks to gain deeper insights into split manufacturing. For instance, our framework indicates that the optimal defense techniques in the two attack scenarios are the same. Moreover, larger benchmarks are secure by naïve split manufacturing and do not require any additional defense technique under our cost model and considered attacks. We also uncover a counter-intuitive finding—an attacker using the network-flow attack should not use all the hints; instead, she should use only a subset.**

***Index Terms*—Hardware security, IP protection, Split manufacturing, Game theory, Routing perturbation.**

I. Introduction

A. Split Manufacturing for Intellectual Property Protection

The Intelligence Advanced Research Projects Activity proposed split manufacturing to prevent intellectual property (IP) piracy during the fabrication of integrated circuits (ICs) [1]. Split manufacturing involves splitting an IC design into two parts: the front-end-of-the-line (FEOL), consisting of transistors and lower metal layers, and the back-end-of-the-line (BEOL), consisting of higher metal layers. To manufacture the chip, the designer ships the FEOL to an off-shore, *untrusted* foundry while she/he fabricates the BEOL at a *trusted*, in-house foundry. Since an attacker in the untrusted foundry does not have access to the complete design, she/he cannot perform IP piracy or overproduce the ICs. Thus, split manufacturing allows the designer to reap the benefits of low-cost production from an off-shore foundry while still maintaining an adequate level of security. Researchers have demonstrated the practicality of split manufacturing by fabricating million-transistor designs [2], while Samsung implemented a 28nm node across Austin and South Korea [1]. Many academic works [3]–[11] have performed security studies on split manufacturing (see [12] for a comprehensive review).

Manuscript received March 23, 2021; revised July 22, 2021 and Sept 21, 2021; accepted October 14, 2021. This article was recommended by Associate Editor R. Schaefer. (*Mark Tressler and Kevin Sipple contributed equally to this work.*) (*Corresponding authors: Vasudev Gohil and Satwik Patnaik.*)

This work was supported in part by NSF under Grant 1822840. A part of this research was conducted using the advanced computing resources provided by Texas A&M High-Performance Research Computing.

Vasudev Gohil, Mark Tressler, Kevin Sipple, Satwik Patnaik, and Jeyavijayan (JV) Rajendran are with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843 USA (email: gohil.vasudev@tamu.edu; satwik.patnaik@tamu.edu; jeyavijayan@tamu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/10.1109/TIFS.2021.3122827

B. Prior Works in Split Manufacturing

Attack techniques. Naïve split manufacturing is prone to attack, as the layout tools used by chip designers prioritize metrics such as power, performance, and area (PPA) over security. An attacker can leverage this bias of layout tools to construct hints that help decipher missing connections [3], [7]. If an attacker can correctly recover all the missing connections, she can determine the complete design, enabling IP piracy. The *proximity attack* [3] was the first to construct and leverage hints. Researchers then advanced it by using various FEOL-level hints as a network-flow problem [7]. These hints are (i) physical proximity of gates, (ii) non-formation of combinational loops, (iii) load capacitance constraints of standard cells, (iv) the directionality of dangling wires, and (v) timing constraints. Other proximity attacks use routing hints (routing proximity, routing congestion, etc.) [5] and utilize machine-learning techniques to recover missing connections [9], [10].

Defense techniques include either: (i) perturbing the locations of gates (placement perturbation) [4], [7], or (ii) perturbing the routing of the nets (routing perturbation) [6], [8], [11]. All defenses attempt to nullify the heuristics used by an attacker (e.g., by diminishing placement and/or routing proximity, by re-routing wires), thereby forcing the proximity attack [3] and the network-flow attack [7] to make incorrect connections. In this work, we consider both placement and routing perturbation techniques—*BEOL+Physical* (BP), *Logic+Physical* (LP), *Logic+Logic* (LL) [7], and randomized insertion of routing blockages [10] (which we denote as RP).[1] We consider these techniques due to their ability to cause high output corruption in an attacker's recovered design. BP selects gates based on connections in the BEOL and perturbs to maximize the physical distance between connected gates (*BEOL+Physical*). LP selects gates with high Hamming

[1]Routing blockages are used to either prevent the routing of wires in a specific area of the chip or prevent specific metal layer(s) from being used.

TABLE I
SUMMARY OF DIFFERENT SCENARIOS AND ATTACK MODELS CONSIDERED IN THIS WORK

Scenario	Attack model	Practical situation	Model type
Scenario #1	Attacker knows the probabilities with which the defender uses different defense techniques.	Defender selects the probabilities based on the efficacy reported in the existing literature. Since the literature is public knowledge, the attacker knows how likely a particular defense technique is to be used.	Sequential
Special case of scenario #1	Attacker knows the exact technique used by the defender.	To find the optimal defense technique under Kerckhoff's principle, i.e., the attacker knows which technique is used by the defender.	Sequential
Scenario #2	Attacker does not know which particular defense technique has been used by the defender.	Attacker is oblivious to the defender's preference for different defense techniques.	Simultaneous

distance (i.e., logical distance) from their neighbors and perturbs to maximize the physical distance between connected gates (*Logic+Physical*). LL selects gates like LP but perturbs to maximize the probability of corrupting the outputs (*Logic+Logic*). RP inserts routing blockages which forces the layout generation tool to re-route wires and avoid specific metal layers, which makes it challenging for proximity attacks to decipher the missing connections [5], [7], [10], [11].

Metrics. Researchers primarily use three metrics to evaluate the success of attack/defense techniques [3], [4], [7], [8], [11]: (i) Hamming distance (HD), which captures the average bit-level mismatch between the outputs of the original and attacker's recovered design under a large (e.g., 50,000) set of input patterns; (ii) output error rate (OER), which indicates the probability of an output being incorrect under a large set of input patterns; and (iii) correct connection rate (CCR), the ratio of connections correctly inferred to the total number of missing connections.[2]

Attacker's capabilities. In this work, we assume the conventional threat model followed by the split manufacturing community. The FEOL foundry is untrusted, and the BEOL foundry and end-user are trusted. The attacker's goal is to recover the missing connections in the FEOL layout—correctly deciphering all the missing connections enables the attacker to pirate and/or overproduce the design. Since the attacker is in the foundry, she/he has access to the technology library and other files used for the generation of the physical layout of the design. Further, the attacker knows (i) the structure of the standard/logic cells, (ii) the size of logic cells, (iii) the input and output capacitance of logic cells, (iv) the associated gate delays of logic cells, and (v) the capacitance of wires in different metal layers. Another important assumption is that the attacker does not know the functionality of the underlying design nor has access to a functional IC, which stems from the threat model adopted by the split manufacturing community.

C. This Work

Optimal defense technique. Intuition suggests a defender should use all the available defense techniques for better security. However, each defense technique amounts to PPA overheads [7], [8], [11]. Also, an attacker has various attack techniques to choose from and the defender does not know which technique will be used by the attacker a priori. Hence, the defender faces these questions: (i) *Will the desired security be achievable under a given PPA cost (overhead) constraint?* (ii) *How successful is a defense technique when the attack technique is unknown?* Moreover, since the attacker knows that the defender aims to achieve a given security level with minimal PPA cost, the attacker can use appropriate techniques to thwart the techniques used by the defender. Note that the existing works in split manufacturing have focused on maximizing security against a particular attack [3]–[11]. However, we aim to find the optimal defense technique (that maximizes security) when the choice of the attack technique is unknown to the defender.[3] The choice of an optimal defense technique presents a non-trivial problem: *find the optimal defense technique under uncertainty in the attacker's choice of attack technique rather than maximize the security against a specific attack technique.* It is important to focus on the case with uncertainty because doing so leads to interesting insights. For instance, we observe that under our cost model and considered attacks, larger benchmarks require no defense technique beyond naïve split manufacturing. In this work, we consider two attack scenarios as highlighted in Table I.

Scenario #1: Here, the defender can randomly choose from the set of available techniques (by assigning probabilities). The defender decides these probabilities based on the efficacy reported in prior works. We assume that the attacker knows the probabilities with which the defender uses different techniques. In this scenario, we do not assume that the attacker knows which specific defense technique has been used; rather, just the probability of any particular defense technique being used. As the prior works are public knowledge, it is straightforward for the attacker to know the probabilities with which the defender will use any particular defense technique. For instance, suppose that out of the three placement perturbation techniques, the defender is thrice as likely to use LP than BP. Similarly, suppose she is four times as likely to use LL than BP. Thus, the probabilities of using BP, LP, and LL would be $0.125, 0.375$, and 0.5, respectively.

Special case of scenario #1: Here, we assume that the attacker knows the exact defense technique used by the defender. This special case is in agreement with the Kerckhoff's principle, i.e., the approach used to protect a certain asset (design IP in our case) is not considered a secret.

Scenario #2: Here, we assume that the attacker does not know which particular defense technique has been used,

[2]Note that the attacker aims to minimize OER and HD whereas the opposite is true for the defender.

[3]We wish to clarify here that we are not assuming that the attack techniques are unknown to the defender. The defender knows all the details regarding various attack techniques but does not know which of those techniques will be used eventually by the attacker.

not even the probabilities with which the defender uses the different techniques. In other words, the attacker only knows what techniques the defender *can* use and nothing more. This scenario covers all attack situations which are not covered by scenario #1. Thus, the two attack scenarios are exhaustive.

Note that in both the scenarios, the defender has no idea about the probabilities of the attack techniques. This is because the defender chooses the defense technique first, and consequently, the attacker chooses the attack technique. The prime focus of this work is to develop a framework which can compare and contrast the optimal defense techniques in the two scenarios. In particular, we are interested in understanding if the attacker knowing the defense technique makes any difference in the optimal defense techniques. Table I summarizes the two scenarios with the attack models, their corresponding situations, and how we model them using game theory (more details about the models are provided in Sec. III-A).

Why not simple engineering decision? The problem of finding an optimal defense technique cannot be solved using simple decision-making about finding a fixed defense technique which is the best response to a given attack technique. This is because the defender does not know which technique will be used by the attacker. For instance, if the defender knew that an attacker would use an attack technique, a, the defender could use a defense technique, d, which is known to provide the highest OER subject to some PPA constraints. However, one cannot assume that the defender knows that the attacker will use the technique a. Moreover, for attack scenario #2, the defender/attacker does not know which particular technique will be used by the attacker/defender. However, it is still reasonable to assume that they both know the techniques available to each other. Hence, a simple decision-making is not possible since there are uncertainties in the choices of techniques for both the defender and the attacker. The next example explains this in detail under attack scenario #2.

Motivational example. Consider the benchmark c6288 with four attack techniques ($\alpha_1, \alpha_2, \alpha_3, \alpha_4$) and four defense techniques (BP, LP, LL, and ND, which stands for no defense). In this example, we use HD to evaluate security. Table II indicates the HD of the attacker's recovered design. α_1 is the proximity attack [3], α_2 adds the capacitance constraint hint to α_1, α_3 adds the dangling wire hint to α_2, and α_4 is the network-flow attack [7]. For example, consider the design protected using BP defense technique and attacked using α_2 for a split layer of M4 (metal layer 4). The value of 47.59 (highlighted in gray in Table II) denotes the HD between the design recovered by an attacker and the original design for that particular defense technique-attack technique combination.

We intend to solve the following problem: *If the defender does not know what attack technique has been used, what is the best technique for the defender?* In other words, we want a defense technique that leads to maximum security for the defender. This is equivalent to finding a row for which each element is the maximum in its corresponding column in Table II. In this example, no such row exists; thus, no defense technique is optimal by itself. Thus, multiple defense techniques should be used with certain probabilities. We want to find these probabilities with which the defender should use any defense to achieve maximum security. We denote these probabilities of using different defense techniques (that maximize the security of the design for the defender) as *optimal defense strategies*. Similarly, the probabilities of using different attack techniques (that lead to the highest success for the attacker) are denoted as *optimal attack strategies*.

TABLE II
HAMMING DISTANCE FOR THE ISCAS-85 BENCHMARK c6288 FOR FOUR DEFENSE AND ATTACK TECHNIQUES FOR A SPLIT LAYER OF M4. ND INDICATES NO DEFENSE

Attack / Defense	α_1	α_2	α_3	α_4
ND	48.5	48.15	48.53	48.45
BP [7]	47.43	47.59	48.23	48.73
LP [7]	48.23	48.35	48.76	48.81
LL [7]	49.54	49.25	48.38	49.04

To solve this problem of finding the optimal defense strategies, *we need a mathematical framework to analyze the different combinations of attack and defense techniques for both attack scenarios*. For this purpose, we leverage *game theory*.

To find a solution that leads to the highest (lowest) security for the defender (attacker), we model the problem as a game (see Sec. II for background on game theory and Sec. III for framework-related details). However, the modeling is non-trivial. We should identify appropriate security and design metrics in split manufacturing and convert the successes (e.g., OER) and costs (e.g., power, timing) into their equivalent economic variables.[4] After modeling the game correctly, we solve for equilibrium—the techniques that form the equilibrium are optimal strategies for the defender and the attacker. For both attack scenarios, the defender maximizes her success by choosing an optimal strategy even if the attacker's strategy is unknown a priori.

D. Our Contributions

1) We develop the first (to the best of our knowledge) game-theoretic framework to obtain optimal defense strategies using nine attack and five defense techniques in split manufacturing.
2) We model real-world scenarios using two different types of games and empirically show the equivalence between the two scenarios in terms of optimal defense strategies.
3) We formulate an economic model that incorporates the security and layout cost metrics in split manufacturing as financial costs (in US Dollars, without loss of generality) for both the defender and the attacker. This model is essential to set up the game correctly.
4) Our proposed framework is not limited to the attack and defense techniques considered in this work. Our framework is generic and one can analyze any existing (or future) attack and defense technique(s).
5) Our framework is flexible and one can model a game according to their notion of success. For example, if a defender desires maximum security irrespective of PPA overheads, the cost functions in our framework can be set to 0 and optimal strategies will be returned accordingly.

[4]The conversion to economic variables is the most natural way to transform different metrics (OER, wirelength, etc.) into the same units for comparison.

6) Extensive experiments on diverse benchmarks help us gain interesting insights. For example, an attacker using the network-flow attack should not use all the hints and no defense technique is necessary for larger, industrial benchmarks under our cost model and considered attacks.

The remainder of the paper is organized as follows. We present a brief background on game theory in Section II. We explain our models of the split manufacturing games, including the utilities of the players and the algorithms used to solve the games in Section III. We present case studies and provide the important results in Section IV. Related work, discussions, and concluding remarks are provided in Section V, Section VI, and Section VII, respectively.

II. Background: Game Theory

Game theory studies the strategic interactions between two or more players. Let $\mathcal{P}$ be the set of players in a general game with $|\mathcal{P}| = N$. Each player $i \in \mathcal{P}$ can perform actions $\mathcal{A}_i$. This leads to *action profiles*, i.e., all possible combinations of actions for all players, $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_N$. Based on the actions played by all players, each player gets some payoff/utility, decided by *utility functions*. Mathematically, $u_i : \mathcal{A} \to \mathbb{R}, \quad \forall i \in \mathcal{P}$ are the N utility functions.

A *strategy* is a rule that a player uses to choose her actions. Strategies can be *pure*, i.e., choose action a, or they can be *mixed*, i.e., choose action a with probability $\mu(a)$ and action b with probability $\mu(b)$. In particular, for player i, we denote μ_i as a probability distribution over $\mathcal{A}_i$. A *strategy profile* represents the strategies of all the players as a tuple, $\mu = (\mu_1, \mu_2, \cdots, \mu_N)$. Under a mixed strategy profile $\mu = (\mu_1, \mu_2, \cdots, \mu_N)$, because of stochasticity in the choice of actions, the expected utility of player i is

$$\mathbb{E}_\mu(u_i(a_1, \cdots, a_N)) = \sum_{a_1 \in \mathcal{A}_1} \cdots \sum_{a_N \in \mathcal{A}_N} u_i(a_1, \cdots, a_N) \cdot \mu_1(a_1) \cdots \mu_N(a_N) \quad (1)$$

where $u_i(\cdot)$ is the utility function for player i. As the players' actions are independent, the joint strategy profile is factored as a product of the individual strategies.

There are different classifications of games, depending on the different perspectives. Based on the interaction between the players, games can be *competitive* or *co-operative*. There can also be *hybrid* games depending on the level of co-operation between the players. Based on the way the players play, games can be classified as *simultaneous* or *sequential*. In simultaneous games, all the players take their actions without knowing the actions chosen by the other players. A simple example of simultaneous games is *Rock-Paper-Scissors*. In sequential games (e.g., *Chess*), one player chooses her actions before the other does.

Another way of classifying games is based on the information available to the players. In *complete information* games, each player is aware of all the actions available to the other players and the corresponding payoffs for all the players. Conversely, in a game with *incomplete information*, a player might know the actions of other players but is not aware of the payoffs of the other players.

III. Game-Theoretic Analysis of Split Manufacturing

In this section, we model the problem of finding optimal defense strategies in split manufacturing using game theory. However, this is not straightforward, as it presents specific challenges, which we discuss next.

A. Challenges in Modeling Split Manufacturing as a Game

Is there a game to begin with? Through several iterations of attacks [3], [5], [7], [9], [10] and defenses [4], [6]–[8], [11], researchers have proposed various techniques in split manufacturing. This enables the attacker and the defender to choose the techniques they want to use or a combination of them. This creates a setting similar to a game where players have multiple actions to choose from. Hence, there exists an underlying game in split manufacturing that can be modeled using game theory.

What kind of a game is it? We refer to the attack scenarios (Table I) to determine the type of the game. For scenario #1, since the attacker knows the probabilities with which the defender uses any defense technique, a sequential game is the correct model. Moreover, since the attacker knows the aforementioned probabilities, the defender needs to commit to a mixed strategy before the attacker chooses her attack technique. Hence, the game is *sequential with a commitment to mixed strategies* by the defender. For scenario #2, since the attacker is oblivious to the choice of defense technique, the scenario should be modeled using a *simultaneous game with mixed strategies*. At first glance, it might seem that for scenario #2, since the attacker chooses her technique after the defender, it should be modeled as a *sequential game*. However, on a closer look, it is clear that since the attacker does not know the choice of the technique used by the defender, the difference in the time of play provides no advantage to the attacker. This necessitates a *simultaneous game* model. Moreover, as the defender aims to protect the design IP from the attacker, there is no cooperation between the players. Hence, the game is *non-cooperative* in both scenarios. In conclusion, we model scenario #1 as a *non-cooperative sequential game with a commitment to mixed strategies* and scenario #2 as a *non-cooperative simultaneous game with mixed strategies*.

We assume that the players are rational and intelligent. Rationality implies that the players aim to maximize their utilities. The assumption of intelligence implies that the players have the same amount of knowledge of the utility functions of both players. Therefore, both games considered in this work are complete information games.

How to model the split manufacturing problem? To model either of the games identified above, we need to identify the players, their actions, and utilities. In split manufacturing, the players are the defender and attacker, and their available actions are the defense and attack techniques. However, *identifying utilities is not trivial*, which we describe next.

How to identify the utility functions of the players? It is essential to define the utility functions of the defender and the attacker such that they capture the economic value to the respective players. For an attacker, the economic value

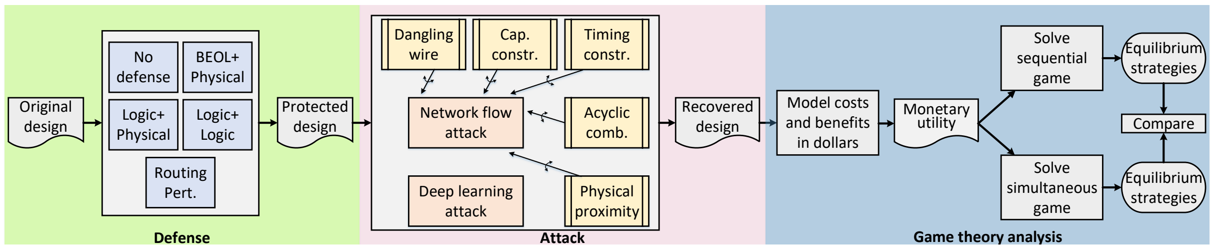


Fig. 1. Proposed framework. The design is protected using any one of the five defense techniques (including no defense). The attacker tries to recover the missing connections in the protected design using different combinations of hints or using the deep learning attack. The recovered design is then evaluated using the metrics (OER and HD). The security and design metrics are translated into economic values by modeling the gains/losses and costs. The resulting utilities are used to solve the sequential and simultaneous game for equilibrium strategies, i.e., optimal attack and defense strategies. These strategies from the two models are compared at the end.

includes the cost of the IP retrieved at the expense of computation resources devoted to the attack technique. Similarly, the economic value for the defender includes the cost of the protected IP at the expense of PPA overhead incurred due to the defense technique. Since there are no notions of calculating these economic values, we need to develop them. More precisely, we must use the metrics in split manufacturing to evaluate the value of a recovered design for the players. The utility functions will provide the economic worth of each attack-defense pair for both the players.

How scalable is the model? For practicality, the game-theoretic analysis needs to be computationally efficient. Since the proposed framework involves performing all combinations of attack and defense techniques on any given design, it is time-consuming (ranging from minutes for smaller designs to several hours for larger designs) to generate the economic values. To ensure feasibility, we need the game solver to quickly return the optimal strategies, i.e., the game solver should not be the bottleneck.

B. Formulating the Split Manufacturing Problem as a Game

Recall that the goal of our model is to solve the problem of finding optimal defense strategies for the two attack scenarios (Table I). We achieve this by setting up games for different benchmarks and solving them for equilibrium. Next, we present the formulation of the games.

Players and their actions. There are two players in the split manufacturing game: the defender in the design house (D) and the attacker in the foundry (A). So, $\mathcal{P} = \{D, A\}$. Each player has a set of actions.

On the one hand, the defender, D, has access to the three placement perturbation techniques [7] and one routing perturbation technique [10]. Additionally, *ND* denotes the inherent defense due to naïve split manufacturing without using any defense technique. We denote the action set by $\mathcal{A}_D = \{\text{ND, BP, LP, LL, RP}\}$.

On the other hand, the attacker, A, tries to identify the missing connections from the FEOL layout. Recall that identifying all the missing connections enables the attacker to perform IP piracy and/or overproduce the design. To that end, we implement the proximity attack [3], which leverages the physical proximity and the acyclic combination hint, and the variants of the network-flow attack [7]. We allow the attacker a choice of using/not using the additional hints (load capacitance constraints, the directionality of dangling wires, and timing constraint).[5] Additionally, we also include the deep learning (DL)-based attack [10].

For the variants of the network-flow attack, any combination of the remaining three hints can be leveraged (Fig. 1). These combinations give rise to eight different attack techniques. Therefore, the action set of the attacker is $\mathcal{A}_A = \{000, 001, 010, 011, 100, 101, 110, 111, DL\}$. In each string, the bits from left to right indicate the choice of the load capacitance constraints, the directionality of dangling wires, and the timing hint, respectively. We denote these combinations as $\{A1, ..., A8\}$. $A1$ (i.e., 000) is the proximity attack [3], $A8$ (i.e., 111) is the network-flow attack [7], and DL is the deep learning-based attack [10].

Both players need to choose a strategy for selecting actions from their action sets. We allow strategies to be mixed. We denote μ_D (μ_A) as the mixed strategy over $\mathcal{A}_D$ ($\mathcal{A}_A$) for the player D (A). We denote the strategy profile as $\mu = (\mu_D, \mu_A)$. Here, μ_D (μ_A) is a probability distribution over the set of actions available to the defender (attacker). In split manufacturing, a mixed strategy implies that the defender (attacker) chooses a defense (attack) technique randomly according to the probability weight of that action given by μ_D (μ_A).

Utility functions need to incorporate both the gain/loss to the player and the cost of the chosen action. However, defining these utility functions is non-trivial for the following reasons: (i) One needs to identify the metrics that affect the utility of each player (e.g., HD, timing, power), (ii) One must translate the security and overhead metrics into appropriate economic values (e.g., US Dollars, without loss of generality), (iii) One needs to ensure that the units of the costs and gains/losses of the players are the same; otherwise, the comparisons would be incorrect, and finally (iv) One needs to identify appropriate functions of these economic variables that constitute the utility of the players in the real world. To this end, we first analyze the gains/losses and then the costs of the players.

C. Transforming Security and Design Metrics into Economic Values

Here, we obtain expressions for dollar values of the players' gains/losses and costs based on the security and design metrics relevant to split manufacturing.

[5]We consider the proximity attack [3] as the baseline since we observe the importance of physical proximity and non-formation of combinational loops. Hence, the switches for these hints should always be on (Fig. 1).

Defender's loss. Let $l_D(\lambda)$ denote the loss to the defender for a recovered design with HD λ. We follow the ideas presented by *Integrated Circuit Engineering Corporation* [13]. The value of the original design is the investment to produce the design, which depends on the engineer-time and other costs. The value of the engineer-time depends on the rate at which an engineer designs netlists, the number of gates, and the labor cost of the engineer. Let r denote the rate (per month) at which an engineer designs gates, z_{lbr} denote the labor cost of the engineer per month, and N denotes the number of gates in a design. Thus, the value of the original design is $\frac{N}{r} \cdot z_{lbr}$. The parameter r is estimated from [13] that provides information about the different facets in IC design (more details in Sec. IV-B).

In addition to the engineer-time, the value of a design also depends on fabrication costs, which depends on the silicon area of the design. Suppose the cost of fabrication for a $1mm^2$ chip is z_{fab} US Dollars. Suppose defense d_j results in a silicon area ζ_{d_j} in mm^2. Then, the corresponding fabrication cost is $\zeta_{d_j} \cdot z_{fab}$ US Dollars. Assuming M chips are fabricated of a given design, the total value of the design is $\frac{N}{r} \cdot z_{lbr} + \zeta_{d_j} \cdot z_{fab} \cdot M$.

Note that the aforementioned model used to calculate the value of the design is not exhaustive. In reality, the value of a design is a function of several variables other than the number of gates in the design, which include factors like ideation, 3rd party IP modules, testing, and architecture-level designing effort. However, we do not consider these factors in our model due to the following reasons:

1) These costs are independent of the actions chosen by the defender or the attacker. They are constants that would be multiplied to all the utility values of the defender in the game matrix. Therefore, mathematically, they would not affect the optimal actions of the players at all.
2) Calculating the exact monetary cost of the design is not a scientific contribution of our work. This is because the values and costs are subject to the judgement of the stakeholder. Our primary goal is to provide a framework that industrial companies and defense agencies can use to decide the best course of action (i.e., optimal defense technique) against IP piracy in split manufacturing.

However, the total value of the design derived above is an estimate for the original design and not for the partially (or fully) recovered design by the attacker. To obtain the loss for a recovered design, we scale the original value with a function of the HD of the recovered design. To that end, we use the exponential loss function $e^{-\alpha \cdot \text{HD}}$. The choice of this function is inspired by [14]. The scaling function should be concave since the returns on additional HD decrease as HD increases. If the HD of a recovered design with N gates is $\lambda\%$, where λ is between 0 and 100, the loss to the defender is $l_D(\lambda) = [\frac{N}{r} \cdot z_{lbr} + \zeta_{d_j} \cdot z_{fab} \cdot M] \cdot e^{-0.1\lambda}$ dollars.[6] The defender aims to minimize this loss, i.e., maximize the difference between the original value and the loss. Mathematically, the defender aims to maximize $[\frac{N}{r} \cdot z_{lbr} + \zeta_{d_j} \cdot z_{fab} \cdot M] - l_D(\lambda) = [\frac{N}{r} \cdot z_{lbr} + \zeta_{d_j} \cdot z_{fab} \cdot M] \cdot (1 - e^{-0.1\lambda})$.

[6]The boundary conditions on the scaling factor are 1 when HD is 0% and close to 0 when HD is close to 100% leads to $\alpha = 0.1$.

Attacker's gain. Let $g_A(\lambda)$ denote the financial gain for the attacker for a given HD (λ) of the recovered design. For the attacker, the absolute value of the recovered design remains the same as derived above, except for a sign change. Since the attacker aims to minimize the HD, we multiply the formula developed above with -1.[7] If HD of a recovered design with N gates is $\lambda\%$, the attacker's gain is $[\frac{-N}{r} \cdot z_{lbr} - \zeta_{d_j} \cdot z_{fab} \cdot M] \cdot (1 - e^{-0.1\lambda})$ dollars.

Defender's overhead cost. Let $c_D(d_j)$ denote the monetary cost to the defender for using defense technique d_j. This technique, d_j, results in timing and power overheads in the design. An increase in timing overhead results in decrease in the maximum operating frequency of the design. Designs operating at lower frequency are worth less money, as evidenced by *Intel*'s binning process [15]. Likewise, an increase in power consumption translates to reduction in the value of the design because of binning.

Let z_{deg_f} denote the percentage drop in the dollar value of the design for 1% degradation in the maximum operating frequency. Let f be the maximum operating frequency of the original design. Suppose defense d_j changes the maximum operating frequency to $f - \delta f_{d_j}$ so that the chip belongs to a different bin. Thus, the percentage degradation in the maximum operating frequency because of a change in the bin of the chip due to frequency overhead of δf_{d_j} is $100 \cdot \left(\frac{\delta f_{d_j}}{f}\right)$. The corresponding percentage drop in the dollar value due to the overhead is $100 \cdot z_{deg\_f} \cdot \left(\frac{\delta f_{d_j}}{f}\right)$. Thus, the cost overhead due to frequency degradation can be obtained by multiplying with the value of the original design as

$$100 \cdot z_{deg\_f} \cdot \left(\frac{\delta f_{d_j}}{f}\right) \cdot \left[\frac{N}{r} \cdot z_{lbr} + \zeta_{d_j} \cdot z_{fab} \cdot M\right] \quad (2)$$

Next, we analyze the cost due to additional power consumption. Let P be the power consumption of the original design. Suppose defense d_j changes the power consumption to $P + \delta P_{d_j}$ so that the chip belongs to a different bin. Let z_{deg_P} be the percentage drop in the dollar value of the design due to 1% degradation in the power consumption. Then, the percentage drop in the dollar value sue to the power overhead is $100 \cdot z_{deg\_P} \cdot \left(\frac{\delta P_{d_j}}{P}\right)$. Thus, the cost overhead due to the power degradation can be obtained by multiplying with the value of the original design as

$$100 \cdot z_{deg\_P} \cdot \left(\frac{\delta P_{d_j}}{P}\right) \cdot \left[\frac{N}{r} \cdot z_{lbr} + \zeta_{d_j} \cdot z_{fab} \cdot M\right] \quad (3)$$

So, the total cost to the defender for defense d_j is

$$c_D(d_j) = \left[\frac{N}{r} \cdot z_{lbr} + \zeta_{d_j} \cdot z_{fab} \cdot M\right] \cdot \left[100 \cdot z_{deg\_f} \cdot \left(\frac{\delta f_{d_j}}{f}\right) + 100 \cdot z_{deg\_P} \cdot \left(\frac{\delta P_{d_j}}{P}\right)\right] \quad (4)$$

[7]An attacker assigns more value for a design with a lower HD (e.g., 10%) since the deviation in terms of functionality from the original design would be less than a design having a higher HD (e.g., 20%). Additionally, researchers in the split manufacturing community have extensively used HD as a proxy for quantifying the security level of a design [3], [7], [8], [11].

Attacker's compute cost. The monetary cost to an attacker for choosing an attack technique a_k is $c_A(a_k)$ which depends on the overall execution time of the technique. We denote the cost of one hour of computing time as z_{comp} (in US Dollars). Assuming the computing time of the technique a_k is t_{a_k} hours, the cost to the attacker is $t_{a_k} \cdot z_{comp}$ dollars.

Note that in addition to the cost for executing the attacks, an attacker might incur costs to develop new attacks, i.e., costs to develop attacks that have not been proposed by the research community. These costs can be modeled using software cost estimation models like COCOMO [16]. However, since we did not formulate/develop the attacks considered in this work, the software development costs to the attacker are 0.

D. Putting it All Together

Having defined the costs and the gains/losses of both the players, we can now define the utilities, i.e., the net dollar values for the different combinations of attack and defense techniques. This will help us select the optimal actions for the attacker and the defender. Suppose an engineer, designing gates at the rate of r, designs a netlist with N gates having an original wirelength l. Let the labor cost of this design be z_{lbr}. A defender uses a defense technique d_j resulting in a frequency overhead δf_{d_j}. This frequency overhead translates into dollar-cost according to z_{deg_f}. Similarly, a power overhead of δP_{d_j} translates into dollar-cost according to z_{deg_P}. The area of the chip after using d_j is ζ_{d_j} which translates into fabrication cost according to z_{fab}. Suppose M chips are fabricated for the design. An attacker uses an attack technique a_k for a certain time t_{a_k} which costs z_{comp} per hour. Let the HD of this recovered design be $\lambda\%$. Thus, the utility of the defender is

$$\begin{aligned} u_D(d_j, a_k) &= \left[\frac{N}{r} \cdot z_{lbr} + \zeta_{d_j} \cdot z_{fab} \cdot M\right] - l_D(\lambda) - c_D(d_j) \\ &= \left[\frac{N}{r} \cdot z_{lbr} + \zeta_{d_j} \cdot z_{fab} \cdot M\right] \cdot \Bigg\{ (1 - e^{-0.1\lambda}) \\ &\quad -100 \cdot z_{deg\_f} \cdot \left(\frac{\delta f_{d_j}}{f}\right) - 100 \cdot z_{deg\_P} \cdot \left(\frac{\delta P_{d_j}}{P}\right) \Bigg\} \end{aligned} \tag{5}$$

and the utility of the attacker is

$$\begin{aligned} &u_A(d_j, a_k) = g_A(\lambda) - c_A(a_k) \\ &= \left[\frac{-N}{r} \cdot z_{lbr} - \zeta_{d_j} \cdot z_{fab} \cdot M\right] \cdot (1 - e^{-0.1\lambda}) - [t_{a_k} \cdot z_{comp}] \end{aligned} \tag{6}$$

Note that we define the utility functions to perform a concrete analysis of the split manufacturing game. A stakeholder can change these functions appropriately to model their costs and gains/losses to derive the corresponding optimal strategies. For instance, in national security applications, the loss to the defender due to the connections recovered by the attacker may completely overshadow the costs to use additional defense techniques. Hence, the utility functions may need to be adjusted accordingly.

Given these utility functions and the stochastic nature of the selection of actions, the expected utilities of the defender and the attacker are

$$U_D(\mu_D, \mu_A) = \sum_{a_k \in \mathcal{A}_A} \sum_{d_j \in \mathcal{A}_D} [u_D(d_j, a_k)] \cdot \mu_D(d_j) \cdot \mu_A(a_k)$$

$$U_A(\mu_D, \mu_A) = \sum_{a_k \in \mathcal{A}_A} \sum_{d_j \in \mathcal{A}_D} [u_A(d_j, a_k)] \cdot \mu_D(d_j) \cdot \mu_A(a_k)$$

Next, we illustrate how to solve the game for the two scenarios.

Scenario #1. Recall that in scenario #1, the attacker observes the mixed strategy (or pure strategy in the special case of scenario #1) of the defender and then chooses her action. In general, the attacker aims to find the action that maximizes her expected utility under uncertainty in the defender's choice of action. Let $b(\mu_D) : \mu_D \to a$ denote the best response for the attacker, i.e., the action a results in maximum utility for the attacker given defender's mixed strategy μ_D. Mathematically,

$$b(\mu_D) = \arg \max_{a_k \in \mathcal{A}_\mathcal{A}} \sum_{d_j \in \mathcal{A}_D} u_A(d_j, a_k) \cdot \mu_D(d_j) \tag{7}$$

Then, a strategy profile (μ_D, b) is an equilibrium (called *Stackelberg equilibrium*) of this game if it satisfies the following:

$$U_D(\mu_D, b(\mu_D)) \geq U_D(\mu'_D, b(\mu'_D)), \quad \forall \mu'_D \tag{8}$$

$$U_A(\mu_D, b(\mu_D)) \geq U_A(\mu_D, b'(\mu_D)), \quad \forall \mu_D, b' \tag{9}$$

$$U_D(\mu_D, b(\mu_D)) \geq U_D(\mu_D, \tau(\mu_D)), \quad \forall \mu_D \tag{10}$$

where $\tau(\mu_D)$ denotes the set of attacker's best responses to μ_D. Here, Eq. (8) ensures that the defender plays the best strategy, Eq. (9) ensures that the attacker plays the best response, and Eq. (10) ensures that the attacker breaks ties in favor of the defender.[8] We can see from these equations that the equilibrium strategies of the players are indeed the optimal strategies in scenario #1 because they lead to the highest utility for both players.

The strategy profile that satisfies the above conditions can be found using linear programming. Algorithm 1 covers the process of finding the optimal (i.e., the equilibrium) strategies. Line 1 in Algorithm 1 iterates over every action available to the attacker. For each action a_k, we solve a linear program (lines 2 to 5). We compute a mixed strategy for the defender such that: (i) playing a_k is the best response for the attacker (line 4), and (ii) the computed mixed strategy maximizes the defender's utility (line 2). These two conditions satisfy the requirements of Eqs. (9) and (8), respectively. The *if* condition (line 6) checks if the above linear program is feasible. Line 7 checks if the value of objective (i.e., the defender's utility) for the current linear program is the highest so far. This ensures that the optimal strategies satisfy Eq. (10). If both conditions are satisfied, the optimal strategies for both players and the optimal utility of the defender are updated.

[8]Note that when the attacker is indifferent among multiple actions, the defender can slightly modify her strategy to ensure that the attacker strictly prefers the action that is optimal for the defender. The tie-breaking requirement here is merely to ensure that the optimal solution is well-defined.

Algorithm 1: Finding optimal strategies for scenario #1

```
Input: $u_D(\cdot,\cdot), u_A(\cdot,\cdot), \mathcal{A_D}, \mathcal{A_A}$
Output: $t^*, p^*$
Initialization: $v^* \leftarrow 0; t^* \leftarrow \phi; p^* \leftarrow \phi$
1  for $a_k \in \mathcal{A_A}$ do
2      maximize $v_k = \sum_{d_j \in \mathcal{A_D}} p_{d_j} u_D(d_j, a_k)$
3      subject to:
4      $\forall a' \in \mathcal{A_A}, \sum_{d_j \in \mathcal{A_D}} p_{d_j} u_A(d_j, a_k) \geq$
           $\sum_{d_j \in \mathcal{A_D}} p_{d_j} u_A(d_j, a')$
5      $\sum_{d_j \in \mathcal{A_D}} p_{d_j} = 1; \quad p_{d_j} \geq 0 \quad \forall d_j \in \mathcal{A_D}$
6      if solution exists then
7          if $v_k > v^*$ then
8              $v^* \leftarrow v_k$;
9              $t^* \leftarrow a_k$;
10             $p^* \leftarrow [p_{d_1}, \ldots, p_{d_{|\mathcal{A_D}|}}]$;
11         end
12     end
13 end
```

Algorithm 2: Finding optimal strategies for scenario #2 using support enumeration

```
Input: $u_D(\cdot,\cdot), u_A(\cdot,\cdot), \mathcal{A_D}, \mathcal{A_A}$
Output: $sols$
Initialization: $sols \leftarrow \phi$
1  for $S_1 \subset \mathcal{A_D}$ and $S_2 \subset \mathcal{A_A}$ do
2      Solve:
3      $\sum_{d_j \in S_1} p_{d_j} u_A(d_j, a_k) = v \quad \forall a_k \in S_2$
4      $\sum_{a_k \in S_2} q_{a_k} u_D(d_j, a_k) = u \quad \forall d_j \in S_1$
5      $v \geq \sum_{d_j \in S_1} p_{d_j} u_A(d_j, a_k) \quad \forall a_k \notin S_2$
6      $u \geq \sum_{a_k \in S_2} q_{a_k} u_D(d_j, a_k) \quad \forall d_j \notin S_1$
7      $\sum_{d_j \in \mathcal{A_D}} p_{d_j} = 1; \quad p_{d_j} \geq 0 \quad \forall d_j \in \mathcal{A_D}$
8      $\sum_{a_k \in \mathcal{A_A}} q_{a_k} = 1; \quad q_{a_k} \geq 0 \quad \forall a_k \in \mathcal{A_A}$
9      if solution exists then
10         $sols.append(([p_{d_j}], [q_{a_k}]))$
11     end
12 end
```

In summary, out of all the actions for which the linear program is feasible, we choose the action (t^* in line 6) which maximizes the objective (v^*) of the linear program as the optimal attack. Further, the defender chooses the probability weights given by the corresponding p^* (line 7). This strategy profile (p^*, t^*) is an optimal (i.e., equilibrium) strategy profile for scenario #1. Note that the linear program may be infeasible for some attacker actions. If an action a_k has less utility for the attacker than another action $a_{k'}$ for all defense actions (i.e., a_k is dominated by $a_{k'}$), then the linear program for action a_k will not be feasible since the constraint in line 4 in Algorithm 1 will be violated. However, there will always be *at least* one attack action for which the program will be feasible.

Recall that in the special case of scenario #1, the attacker knows *exactly* which defense has been used (Table I). Then, the optimal actions can be found easily using the *backward induction algorithm*. The essence of the algorithm is as follows. First, for each defense action, we find the best response action for the attacker, i.e., the action that maximizes the utility for the attacker. Next, from the set of best responses for the attacker, we pick the defense action for which that attack-defense combination gives the highest utility to the defender. An example of this algorithm is presented in Sec. IV. Moreover, please note that the optimal strategies obtained in this special case of scenario #1 will still satisfy Eqs. (8), (9), and (10) as those equations hold for the general setting.

Scenario #2. Recall that, in this scenario, both players are oblivious to the choice of actions of the other player (Table I). In other words, the setting is that of a simultaneous game. Here, a strategy profile (μ_D, μ_A) is an equilibrium (called *Nash equilibrium*), if

$$U_D(\mu_D, \mu_A) \geq U_D(\mu'_D, \mu_A) \quad \forall \mu'_D \tag{11}$$

$$U_A(\mu_D, \mu_A) \geq U_A(\mu_D, \mu'_A) \quad \forall \mu'_A \tag{12}$$

Eq. (11) and Eq. (12) ensure that the defender and the attacker play their best responses, respectively. A strategy profile satisfying these equations can be found using the support enumeration algorithm (Algorithm 2). The algorithm enumerates all combinations of all subsets of the actions (called *supports*) available to both players (line 1). For each combination, we solve a set of linear equations. Lines $3, 4, 5$, and 6 ensure that we have the best responses from both players. Lines 7 and 8 normalize the weights of actions, while lines 9 and 10 add the optimal weights to the list of solutions (if a solution to the above equations exists). We refer the interested readers to [17] for further details.

It is evident from Eqs. (11) and (12) that the strategies constituting a Nash equilibrium is the optimal strategy for the players since they maximize the utilities of the players. Thus, following such a strategy, the defender and the attacker are expected to be the most successful in scenario #2. Moreover, since the split manufacturing game is a finite game[9] with the possibility of mixed strategies, [18] proves that optimal strategies always exist.

In the real-world setting, an attacker would come across an unknown split design, and the attacker evaluates the optimal action as follows. The attacker performs the game-theoretic analysis on a set of known benchmarks. Such an analysis helps the attacker gain insights and they can use the insights to decide which hints/attack techniques work best depending on the type of design. Note that an attacker (in the real-world setting) can launch the attacks but cannot compute the HD since the attacker does not have access to a working chip.

Extending to other split manufacturing metrics. So far, we have used the HD metric, but one can also use the other metrics for split manufacturing, such as OER and CCR. If we use OER, we need to plug the OER values instead of HD in

[9]A game with a finite number of actions and players who play only once.

TABLE III
GAME MATRIX FOR c6288 BENCHMARK. ENTRIES FOR EACH COMBINATION REPRESENT THE PAIR (DEFENDER'S UTILITY, ATTACKER'S UTILITY) I.E., $(u_D(d_j, a_k), u_A(d_j, a_k))$ FOR THE OER METRIC. THE HIGHLIGHTED CELL (IN PURPLE) CORRESPONDS TO THE OPTIMAL STRATEGIES IN SCENARIO #2

Defense \ Attack	$A1$	$A2$	$A3$	$A4$	$A5$	$A6$	$A7$	$A8$	DL
ND	(1696.14, -1696.1616)	(1696.14, -1696.1613)	(1696.14, -1696.1630)	(1696.14, -1696.1614)	(1696.14, -1696.1636)	(1696.14, -1696.1643)	(1696.14, -1696.1627)	(1696.14, -1696.1642)	(1696.14, -1696.1422)
BP	(1695.65, -1696.1621)	(1695.65, -1696.1634)	(1695.65, -1696.1635)	(1695.65, -1696.1621)	(1695.65, -1696.1624)	(1695.65, -1696.1639)	(1695.65, -1696.1627)	(1695.65, -1696.1640)	(1695.65, -1696.1422)
LP	(1695.67, -1696.1619)	(1695.67, -1696.1636)	(1695.67, -1696.1632)	(1695.67, -1696.1640)	(1695.67, -1696.1623)	(1695.67, -1696.1614)	(1695.67, -1696.1630)	(1695.67, -1696.1612)	(1695.67, -1696.1422)
LL	(1695.80, -1696.1638)	(1695.80, -1696.1632)	(1695.80, -1696.1606)	(1695.80, -1696.1622)	(1695.80, -1696.1642)	(1695.80, -1696.1641)	(1695.80, -1696.1624)	(1695.80, -1696.1640)	(1695.80, -1696.1422)
RP	(1696.12, -1696.5209)	(1696.12, -1696.4831)	(1696.12, -1696.3017)	(1696.12, -1696.2980)	(1696.12, -1696.2556)	(1696.12, -1696.2526)	(1696.12, -1696.1661)	(1696.12, -1696.1592)	(1696.12, -1696.1422)

the utility functions. For CCR, we need to change the signs of the gains/losses to the players since the defender aims to minimize the CCR while the attacker seeks to maximize it. *Please note that OER is a stricter metric compared to HD for the attacker*, which is explained next. Even if one output bit is incorrect for an input pattern, that input pattern is classified as a failure. Such a definition has an impact on the optimal strategies, which we discuss next.

IV. RESULTS

A. Experimental Setup

We implement and evaluate our game-theoretic framework (Fig. 1) on ten selected combinational benchmarks from the ISCAS-85 and ITC-99 suites. In addition, we also validate our inferences on four designs from the EPFL-15 benchmark suite [19]. We implement the placement perturbation defense techniques (BP, LP, and LL) [7] in *Python 3.6* and consider no defense (ND) to represent the naïve split-manufactured design. We implement routing perturbation defense (RP) [10] using in-house tool command language (TCL) scripts working within *Cadence Innovus*. We obtained the source codes for the DL attack from [20], [21]. The DL model was trained using a Linux machine with Intel 2.4 GHz CPUs and an NVIDIA Tesla K80 GPU for 24 hours. We implement the proximity attack [3] and all the variants of the network-flow attack [7].

Without loss of generality, we use *Cadence Innovus 19.13* for incremental routing of the perturbed designs. All our physical layouts (both protected and unprotected) are also DRC-clean. We apply 50,000 random input patterns to calculate HD and OER using *Synopsys VCS*. After collecting the security metrics, we solve for optimal strategies. For scenario #1, we solve for the optimal strategies by implementing Algorithm 1 in *Python 3.6*. The linear program is solved using the *Pulp* library package. For the special case in scenario #1, we implement the backward induction algorithm in *Python 3.6*. For scenario #2, we use *Nashpy*, a *Python* library package that solves two-player games using the support enumeration algorithm (Algorithm 2). As mentioned before, we require the model to return the solution quickly (for scalability), and our models return the solutions of the games instantly (< 3s).

For the sake of completeness, we explain the experimental approach here. For every defense-attack combination, the defender generates a protected design using a defense technique. This protected design is then split, and the FEOL part of the design is provided to the attacker. The attacker attacks the protected design using an attack technique of choice and obtains a recovered design. We apply $50,000$ random input patterns to the recovered design and the original design, obtain the corresponding outputs, and calculate the HD of the recovered design. Next, we calculate the utilities of the defender and attacker for the obtained HD using Eqs. (5) and (6). Finally, given the utilities of the defender and attacker, we solve the games using the algorithms (Algorithm 1, Algorithm 2, and backward induction) and obtain the optimal strategies.

B. Parameters for Gains and Costs

Computing the gains. Recall that r denotes the rate (gates per month) at which an engineer designs the netlist. We use the values provided in [13] and extrapolated it to 2020. Based on this analysis, $r = 20,000$ gates/month/person. An IC designer's average annual salary is $\$101,966$ [22]. We scale this value by a labor wrap rate[10] of 2.1 and consider a 2080-hour work year to obtain z_{lbr} as $\$17,842.59$ [23].

We calculate the cost of fabrication for a $1mm^2$ chip, z_{fab} as follows. The cost of a $300-mm$ silicon wafer is \$2274 [24]. In this $300-mm$ wafer, we can fit 70019 $1mm^2$ dies completely.[11] The die yield is assumed to be 0.7 based on the current TSMC die yield of 0.8 for the 5nm node [25].[12] So, $z_{fab} = \frac{2274}{70019\times0.7} = \$0.046/mm^2$. We assume M = 100,000.

Computing the costs. We compare the prices of *Intel* i9 and i7 processors across seven bins to estimate the degradation in their price for degradation in frequency (z_{deg_f}) and power (z_{deg_P}). On average, *Intel* reduces the chips' price by 10.79% for a 29.3% degradation in the clock frequency. Similarly, they reduce the chip's price by 10.79% for a 48% degradation in power. Thus, z_{deg_f} is 0.368% and z_{deg_P} is 0.224%. We calculate the computation cost for the attacker (z_{comp}) using the Amazon Web Services pricing plan [26]. A machine with 16GB memory and 8vCPU is sufficient to perform the attack. Thus, one can use the *a1.2xlarge* instance to perform the attack. Hence, $z_{comp} = \$0.204$/hour.

C. Case Study

Here, we consider two examples for the attack scenarios and visually explain how we solve the games. We consider scenario #2 followed by the special case of scenario #1.

1) Scenario #2: Consider the ISCAS-85 benchmark c6288, protected using five defense techniques and attacked using nine attack techniques (Table III). The success metric considered here is OER. Eqs. (5) and (6) govern the utilities of the players. We perform nine attack techniques on five defense techniques and calculate 45 utilities for each player. These utilities are shown as tuples in Table III. The first/second entry

[10]The labor wrap rate is a constant used to account for other overhead costs and licensing costs of the electronic design automation tools.

[11]Obtained by dividing the area of the wafer by the area of a single chip and subtracting the number of square chips which do not fit entirely on the circular wafer.

[12]The yield is to account for the fact that there will be dies/chips which will be defective.

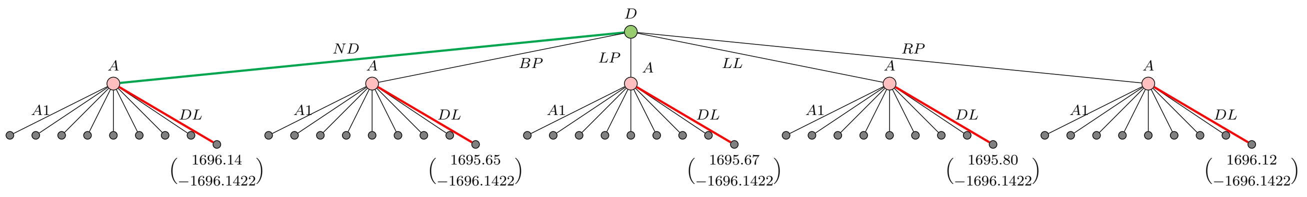


Fig. 2. Representation of the sequential game for the ISCAS-85 benchmark c6288 in the special case for scenario #1. The root node (D) indicates that the defender plays first, followed by the attacker (A) in the next level of the tree. The edges represent the actions (ND, BP, LP, LL, RP for D and $A1, \ldots, A8, DL$ for A). The values for leaves (where indicated) represent the utilities of the defender and the attacker, in that order from top to bottom. The utilities for the other leaf nodes are not shown for readability. However, the values can be obtained from Table III.

of each tuple corresponds to the defender's/attacker's utility. Recall that both players aim to *maximize* their utilities.

As discussed, calculating the equilibrium strategies of the game requires solving linear equations (see Algorithm 2). However, in this game, we can infer the optimal strategies by visual inspection using *iterated dominance*, which is explained next. Since the defender aims to maximize her utility, if there exists any defense technique, $D1$, which has lower utility than any other defense technique, $D2$, then $D2$ *dominates* $D1$, i.e., the defender would never choose $D1$. In this game, we observe that the defender's optimal strategy is ND because ND dominates all other defense techniques.

Since ND is optimal, we can eliminate the rows for BP, LP, and LL from consideration. Thus, the game simplifies to the top row only. To find the optimal strategy for the attacker, we need to find the column corresponding to the maximum utility for the attacker in the ND row. We observe that action DL is optimal (the corresponding cell is highlighted in purple in Table III). Thus, the optimal strategies for this game are (ND, DL) for the defender and attacker, respectively. This example illustrates an important point: *Routing perturbation might not always be required for securing a design. In particular, no defense (ND) can be an optimal technique.* This inference is not specific to the c6288 benchmark and we validate our observations across different benchmarks next.

2) Special case of Scenario #1: We consider c6288 with OER as the success metric. For the sequential setting, we represent the game as a tree, as illustrated in Fig. 2. Since the defender chooses her action first, the root node (D) corresponds to the defender. Each edge from D represents an action available to her. For each action of the defender, the attacker has eight actions to choose from. They are shown as edges from the nodes corresponding to the attacker, A.

Recall that in this special case, the attacker knows which defense technique has been used by the defender. Hence, we can solve it using backward induction, as explained next. The utility values are still the same as in Table III. We can observe that the attacker has a corresponding best response for every action taken by the defender. We highlight these best response edges in red. The utility values are shown at the leaves corresponding to the best responses from the attacker. Thus, the top entry is the defender's utility, and the bottom entry is the attacker's utility. The utilities for the other leaves are not illustrated; however, they can be inferred from Table III.

Based on the utilities for the attacker's best responses (red edges) in Fig. 2, we see that the defender's utility is maximized for action ND (green edge). Also, the attacker's corresponding

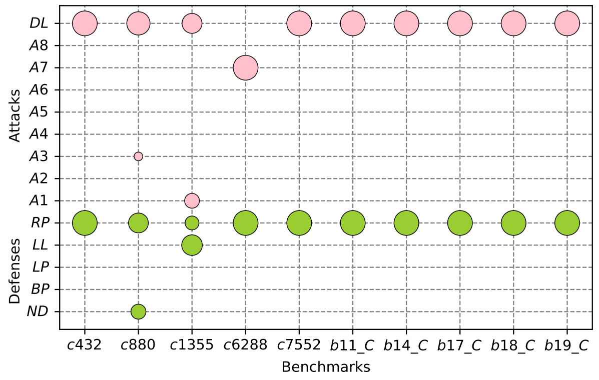


Fig. 3. Optimal strategies for the HD metric in scenario #2.

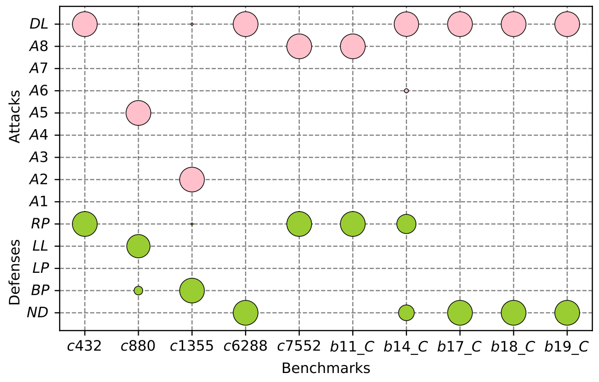


Fig. 4. Optimal strategies for the OER metric in scenario #2.

best response is DL. Hence, (ND, DL) is the optimal strategy profile. We can also verify that this optimal strategy also satisfies Eqs. (8), (9), and (10). Next, we discuss our results for all scenarios for the HD and OER metrics.

D. Optimal Strategies

1) Scenario #2: Here, we analyze the optimal strategies obtained for scenario #2 for ten benchmarks. We consider two metrics: HD and OER. In both the cases, the utility functions remain the same (i.e., Eqs. (5) and (6)), only the value of λ differs. Figs. 3 and 4 display the optimal defense and attack strategies for the HD and OER metrics, respectively. For each benchmark, the size of a green bubble indicates the optimal probability of using that defense technique compared to other defense techniques. Likewise, the size of the red bubble indicates the optimal probability of using that attack technique compared to other attack techniques.

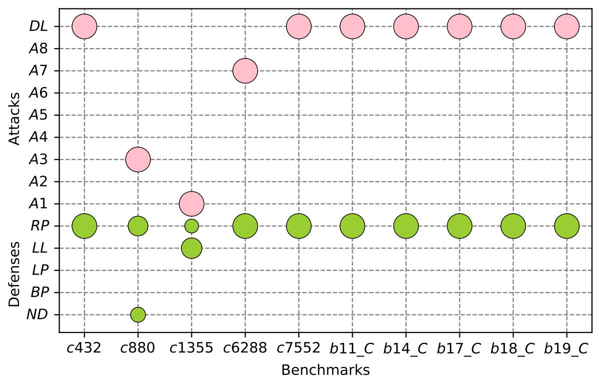


Fig. 5. Optimal strategies for the HD metric in scenario #1.

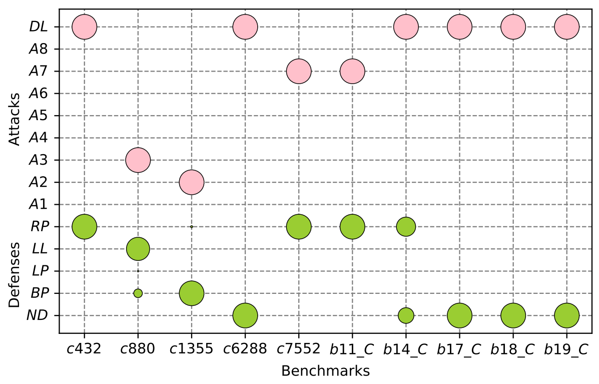


Fig. 6. Optimal strategies for the OER metric in scenario #1.

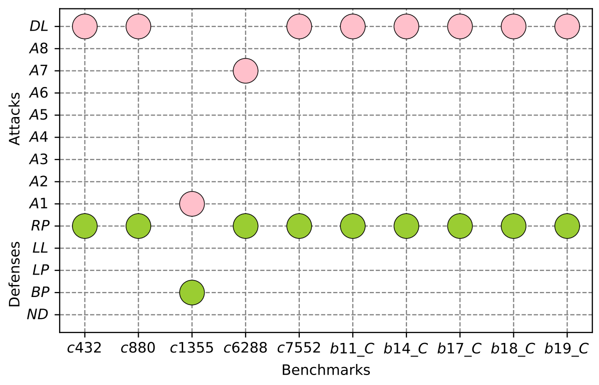


Fig. 7. Optimal strategies for HD metric for the special case in scenario #1.

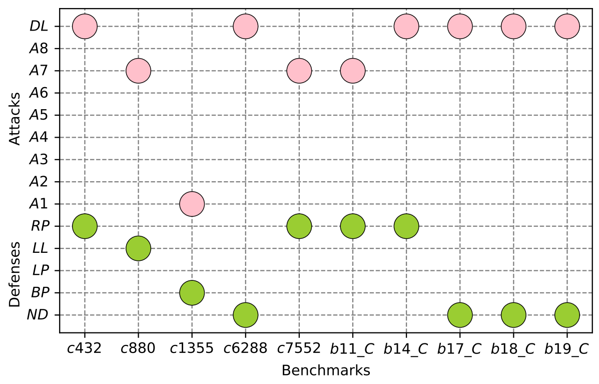


Fig. 8. Optimal strategies for OER metric for the special case in scenario #1.

HD metric. For all benchmarks except c880 and c1355, RP has unit weight, i.e., it is the optimal defense strategy. Even for c880 and c1355, RP is a part of the optimal strategy. This is because RP causes a significant increase in HD while incurring minimal PPA overheads. These observations also validate the fact that RP outperforms the placement perturbation defense techniques. Another inference from the plot is that the DL attack is optimal (almost always) validating that it is better than the network-flow attack.

The **OER metric** demonstrates interesting results. We observe that for smaller benchmarks like c432 and b11_C, RP is still the optimal defense strategy, but for larger benchmarks (c6288, b17_C, b18_C, b19_C), the optimal strategy is ND. Our results indicate that under our cost model and considered attacks, larger benchmarks are inherently secure by naïve split manufacturing and do not require any additional defense technique. This change in optimal strategy for larger benchmarks is because OER is a stricter metric for the attacker.

2) Scenario #1: In this subsection, we analyze the optimal strategies for scenario #1. The optimal strategies are plotted in Figs. 5 and 6. Interestingly, the optimal defense strategies obtained in scenario #1 are the same as the optimal defense strategies for scenario #2 for both OER and HD metrics. This can be seen by comparing the optimal defense strategies in Figs. 3 and 5, and Figs. 4 and 6. This result means that *the defender is not at a disadvantage if the attacker knows the probabilities with which the defender chooses her actions.*

3) Special case in scenario #1: In light of the above result, we ask the following question: *Do the optimal defense strategies change if the attacker knows exactly (not just probabilistically) what defense technique has been used?* In other words, do the optimal defense strategies for the special case in scenario #1 differ from the optimal defense strategies in scenario #1? Implementing the backward induction algorithm for all benchmarks for the HD and OER metrics, we get the optimal actions as illustrated in Figs. 7 and 8.[13]

Comparing these results with Figs. 5 and 6, almost always (except for the HD metric for c1355), the optimal actions for the defender are those actions that have the highest probability in the optimal defense strategies in scenario #1. It makes sense for the defender to capitalize on those actions with a larger weight in the probabilistic setting since those actions are optimal in the more general setting.

E. Impact of Different Utility Functions on Optimal Strategies

Recall that when determining the value of a recovered design (Sec. III-C), we considered the exponential scaling factor $1 - e^{-0.1\lambda}$, where λ is the HD/OER of the recovered design. We chose this function because of continuity, differentiability, and concavity. However, we can also consider other functions that have these properties. In particular, we focus on a family of functions that is represented by

$$f_\eta(x) = \gamma \cdot \frac{(\kappa \cdot x + 1)^{1-\eta} - 1}{1 - \eta} \tag{13}$$

[13]Note that we denote optimal actions instead of optimal strategies to emphasize the fact that the solutions in this case, will always be pure strategies, i.e., actions, not mixed strategies.

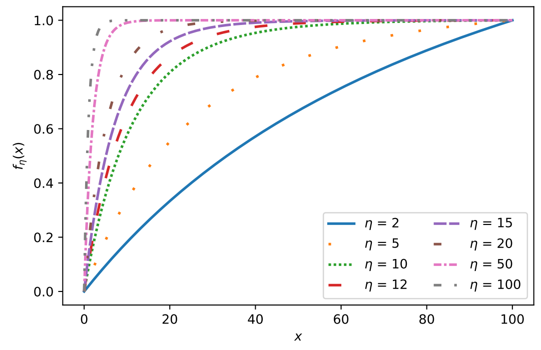


Fig. 9. Family of functions with which the original values of the designs are scaled for analyzing the impact of different functions in the family on the optimal strategies for the defender.

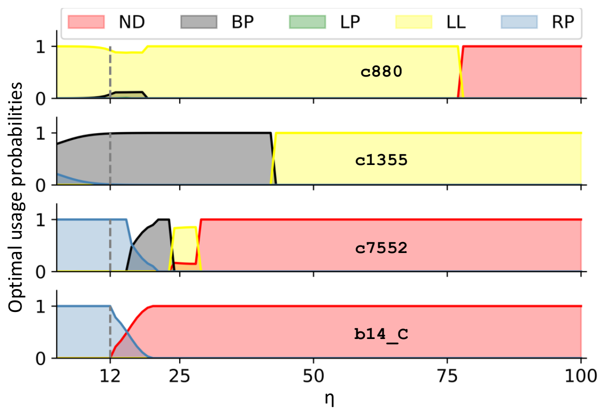


Fig. 10. Optimal strategies for scenario #1 for the OER metric for different values of η.

for $\eta \geq 2$. To satisfy the boundary conditions $f(0) = 0$ and $f(100) = 1$,[14] the values of the parameters should be $\kappa = 0.01$ and $\gamma = \frac{1-\eta}{2^{1-\eta}-1}$. Substituting these values, we get

$$f_\eta(x) = \frac{(0.01 \cdot x + 1)^{1-\eta} - 1}{2^{1-\eta} - 1} \tag{14}$$

To study the impact of other functions on the optimal strategies, we replace the term $(1 - e^{-0.1\lambda})$ in Eqs. (5) and (6) with $f_\eta(\lambda)$ according to Eq. (14). Here, the parameter η controls how close a curve is to the step function. As shown in Fig. 9, as η increases from 2 to 100, the curves approximate the step function better. The slopes of the curves in this family indicate the amount of impact that unit change in HD/OER has on the value of the design. For example, for $\eta = 100$, the slope is very high initially, and then it becomes very close to 0. This indicates that for this scaling factor, even a minuscule HD/OER value leads to no loss to the defender. In other words, for $\eta = 100$, the defender is happy with some small but non-zero value of HD/OER, and she/he does not gain much by further increasing HD/OER. On the other hand, for $\eta = 2$, any connection recovered by the attacker would result in a significant loss to the defender because the scaling factor ($f_\eta(x)$ in the plot) drops from 1 rapidly. This function would be appropriate if the defender does not want to sacrifice security for less PPA overheads.

[14]The boundary conditions remain the same for the attacker and the defender because the sign in the dollar values in Sec. III-C takes care of the two players' opposing objectives. Hence, the scaling factor remains same.

TABLE IV
IMPACT OF SPLIT LAYER ON THE ITC-99 BENCHMARK `b11_C` FOR OER METRIC. THE NUMBERS CORRESPOND TO THE PROBABILITIES OF USING THE DEFENSE TECHNIQUES

Defense \ Split layer	M3	M4	M5
ND	1.0	0	0
BP	0	0	0
LP	0	0	0
LL	0	0	0
RP	0	1.0	1.0

We seek to address the following questions for all benchmarks: (i) For this family of functions, do the optimal strategies remain the same for all values of η? (ii) If not, under what conditions do they change, and how do they vary? (iii) What is the limiting behavior of the optimal strategies?

Fig. 10 illustrates the trend of the optimal defense strategies $\eta \in \{2, 3, \ldots, 100\}$. For `c432` and `b11_C`,the optimal strategy is RP for all values of η, and for `c6288, b17_C, b18_C,` and `b19_C`, the optimal strategy is ND for all values of η. Hence, they are not plotted. We observe that the optimal strategies change with η for some benchmarks. The limiting behavior is also interesting. As η increases, after a certain point, the optimal strategies for `c880, c7552,` and `b14_C` become ND. This happens because as η increases, the additional cost for a unit increase in power and frequency overheads become larger. So, the defender's utility function prioritizes lower PPA overheads over higher security. Hence, ND (with no PPA overheads) becomes the optimal strategy.

F. Impact of Split Layer on Optimal Defense Strategies

As the success metrics depend on the split layer [7], [8], [11], we expect the choice of the split layer to affect the optimal strategy. When the defender opts for a higher split layer, the success for the attacker increases since fewer missing connections need to be recovered. Consider the ITC-99 benchmark `b11_C`. For scenario #2, we observe that splitting after M3 results in ND being the optimal strategy, whereas splitting after M4 and M5 results in RP being the optimal defense strategy. (Table IV). This is because there is no requirement for a defense for lower split layers (M3) since there are many missing connections for the attacker to recover. However, for higher split layers (M4 and M5), we need to rely on defense techniques (e.g., routing perturbation). Similar results also hold for scenario #1.

G. Relevance of Attack Hints and Efficacy in terms of CCR

Recall that the attacker has a total of five hints in the network-flow attack, (i) physical proximity, (ii) non-formation of combinational loops, (iii) load capacitance constraints (Cap.), (iv) directionality of dangling wires (Dang.), and (v) timing constraints (Tmng.). In order to evaluate how well each hint works for the attacker, we calculate the CCR when the attacker uses different combinations of these hints. To get a functional design, we always use the hints of physical proximity and the non-formation of combinational loops.

TABLE V
COMBINATIONS OF ATTACK HINTS WHICH LEAD TO THE HIGHEST CORRECT CONNECTION RATE (CCR) FOR ALL FIVE DEFENSES FOR ALL BENCHMARKS

Defense	ND			BP			LP			LL			RP		
Hint / Benchmark	Cap.	Dang.	Tmng.	Cap.	Dang.	Tmng.	Cap.	Dang.	Tmng.	Cap.	Dang.	Tmng.	Cap.	Dang.	Tmng.
`c432`	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
`c880`	✓	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗
`c1355`	✓	✓	✗	✗	✗	✗	✓	✓	✓	✓	✓	✗	✗	✗	✓
`c6288`	✓	✗	✓	✓	✗	✗	✓	✓	✓	✗	✗	✗	✓	✗	✗
`c7552`	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗
`b11_C`	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗	✓	✗	✗
`b14_C`	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓	✓	✗	✗
`b17_C`	✓	✗	✓	✓	✗	✓	✓	✗	✗	✓	✗	✗	✓	✗	✗
`b18_C`	✓	✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗
`b19_C`	✓	✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗

We calculate the CCR for eight attacks (three hints) for five defense techniques (ND, BP, LP, LL, and RP). One combination (out of eight attacks) would lead to the highest CCR for the attacker (ties are broken in favor of the lowest number of hints used in the order: Cap., Dang., Tmng.). We indicate the successful attack combination in Table V for each benchmark for all the five defense techniques. Consider the ISCAS-85 benchmark `c6288` defended using ND, i.e., no defense. An attacker achieves the highest CCR when load capacitance constraints are used, directionality of dangling wires are not used, and timing constraints are used (in addition to using the physical proximity and the the non-formation of combinational loops). We list the important observations from Table V as follows.

- Across all defense techniques, the load capacitance constraint is the most useful of the three hints, followed by the timing constraint hint.
- In the best combination for the attacker across all defense techniques, whenever the timing constraint hint is used, almost always, the load capacitance constraint hint is also used. This indicates that the timing constraint hint (by itself) is not a powerful hint. The same holds for the directionality of dangling wires hint too.
- There are several instances where the attacker does not require the remaining three hints to achieve the highest CCR. This implies that the physical proximity and the non-formation of combinational loops hints are quite powerful when used together. Note that in such instances, the greedy proximity attack works better than the network-flow attack.

In addition to the game-theoretic modeling and analytical results presented in the prior sections, we also provide some insights about the split manufacturing aspect of the problems as they would help a reader gain an understanding of the purpose and context of the results. In Table VI, we show the CCR, OER, and HD for two attack-defense combinations: RP defense technique against DL attack, and RP defense technique against the $A8$ attack (network flow attack with all hints). The table clearly demonstrates that the DL attack performs significantly better than the $A8$ attack for all considered benchmarks. Moreover, the trend with different benchmarks indicates that the attack success (for both DL and $A8$ attacks) decreases for the larger benchmarks.

TABLE VI
CCR, OER, AND HD FOR RP DEFENSE WITH DL AND NETWORK FLOW ATTACK ($A8$)

Metric	Comb.	`c432`	`c880`	`c1355`	`c6288`	`c7552`	`b11_C`	`b14_C`	`b17_C`	`b18_C`	`b19_C`
CCR	**RP&DL**	55.5	31.8	18.7	13.2	20.7	26.8	13.6	9.3	16.6	11.4
	RP&A8	22.2	14.4	6.0	3.6	9.0	7.6	2.7	1	5.2	4.6
OER	**RP&DL**	29.03	87.33	0	100	99.99	99.94	100	100	100	100
	RP&A8	44.1	98.89	99.99	100	100	100	100	100	100	100
HD	**RP&DL**	12.26	6.73	0	49.92	8.22	20.87	16.41	12.31	5.04	7.45
	RP&A8	18.85	16.7	25.81	49.61	19.75	29.81	27.1	38.18	12.06	50

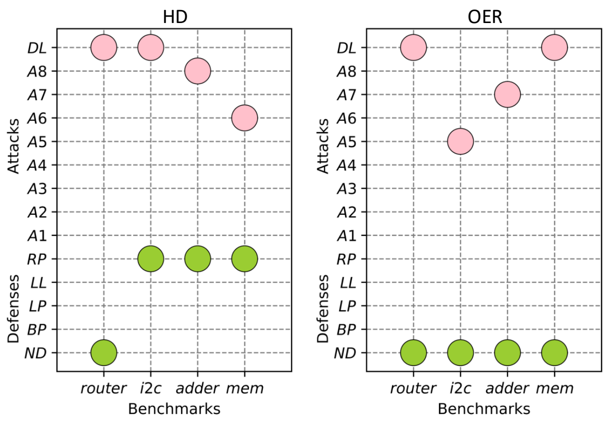


Fig. 11. Optimal strategies for the HD and OER metrics in all scenarios.

H. Validation on New Benchmarks

Here, we show that our general inferences hold true for a set of different benchmarks. In particular, we perform our game-theoretic analysis (for ND, BP, LP, and LL defense actions, and $A1$ to $A8$ attack actions) on the `lookahead XY router`, `i2c controller`, `adder`, and `memory controller` from the EPFL-15 benchmark suite [19]. We plot the optimal strategies for the HD and OER metrics in Fig. 11. We plot the optimal strategies only once for each metric because the optimal strategies are the same for all three scenarios. We list the important inferences from the updated results as follows.

- For the OER metric, ND is the optimal choice for the defender for all the benchmarks. Such an inference is in line with the results on the original ten benchmarks.
- For the HD metric, using the RP defense does not increase the security of the small benchmark, `router`. Hence, ND is the optimal choice for the defender. However, for

TABLE VII
COMPARISON OF OPTIMAL DEFENSE STRATEGIES FOR THE HD METRIC FOR THE ITC-99 BENCHMARK `b11_C` SPLIT AFTER M4

	Defense							
Optimal strategy	**ND**	**BP**	**LP**	**LL**	**RP**	**RP+BP**	**RP+LP**	**RP+LL**
Before	0	0	0	0	1.0	—	—	—
After	0	0	0	0	1.0	0	0	0

the larger designs (`adder`, `i2c`, and `mem`), RP provides better security with minimal overheads, and therefore, RP is the optimal defense technique for the defender.

I. Mixing Defense Techniques

Here, we explore how different defense techniques can be "mixed" and how one can evaluate them in terms of optimality. We mix the placement and routing perturbation techniques using the BP, LP, and LL defense techniques on a layout protected using the RP defense technique. We call these mixed defenses BP+RP, BP+LP, and BP+LL, respectively. We implement the RP+BP, RP+LP, and RP+LL techniques in addition to the individual defense techniques for the ITC-99 benchmark `b11_C` split after metal layer M4. We measure the efficacy (based on the HD metric) of the mixed defense techniques by comparing the optimal strategies when we consider only the individual techniques and the case when we consider both individual and mixed defense techniques. The results of this comparison are shown in Table VII. The row "Before" is for the former case (individual only), and the row "After" is for the latter case (individual and mixed).

The results demonstrate that even after including the mixed defense techniques in the game-play, the optimal defense technique is still RP. This is because RP, by itself, provides sufficient security to the design. The HD values against the optimal attack action, DL, for RP, RP+BP, RP+LP, and RP+LL are $20.87\%, 17.47\%, 16.39\%$ and 15.47%, respectively.

Please note that our observations do not imply that defense techniques, in general, can never be improved by mixing individual techniques. Instead, we demonstrate an example of how our game-theoretic framework can be used to evaluate the overall efficacy of new defense techniques (which stem from mixing/combining different defense techniques).

V. Related Work

Researchers have used game theory to analyze the detection of hardware Trojans [27]–[30]. [27] used a two-person game to model the problem with a hypothetical case study. They only consider pure strategies; however, this assumption has the following issue. An equilibrium, i.e., an optimal strategy, is not guaranteed in this case, and the game provides no information about optimal actions to either of the players. Moreover, thorough experimentation on benchmarks is also missing. This limits the generality and scalability of the results. [28] improves upon [27] by considering mixed strategies; however, they consider only a simultaneous model, which does not cover the entire spectrum of attack scenarios. [29] used a zero-sum model rather than a generic non-zero-sum model. However, this assumption does not hold in real-world scenarios. In reality, the defender incurs overheads to detect Trojans, which need to be accounted for in the value for the defender. [30] only modeled using a game where the defender plays first; the attacker observes the defender's strategy and makes her move. This model does not reflect the real-world setting always because the attacker can learn about the defender's preferences from prior experiences. They also assume a zero-sum setting, which, as mentioned previously, does not model the real-world setting accurately.

VI. Discussion

Why different scenarios? Researchers developing attack techniques in the split manufacturing community typically assume that the attacker has all the details regarding the defense technique implemented by the defender. Recall that the special case in scenario #1 reflects the aforementioned assumption (Table I). Also, even if the attacker is unaware of the *exact* defense technique, she/he can use prior knowledge, public information, or even the information from the physical layout to deduce the probabilities of different defense techniques. Note that scenario #1 covers this situation. However, in a real-world setting, although it might be possible that the attacker can infer the defense techniques or the probabilities of different defense techniques from the physical layout, she/he might not have a reasonable amount of confidence in those inferences. Therefore, we consider scenario #2 separately, where we assume that the attacker does not know the defense technique used by the defender. Considering all the scenarios also leads to a complete treatment of the subject matter. In the (less likely) case that the attacker does not know the defense technique or the probabilities, our observations from scenario #2 can guide the defender on picking optimal actions (optimal defense technique). We summarize the key observations in Table VIII for both the scenarios.

Formal analysis of optimal strategies. Note that researchers have proved the existence of different kinds of equilibria for different games [18], [31], [32]. In particular, researchers have analyzed zero-sum games in great detail [33]. However, since the split manufacturing game is not zero-sum, we can not apply those analyses to our work. An extension of zero-sum games, called strategically zero-sum games, has also been developed [34]. A game (with the players' utility matrices denoted by A and B) is strategically zero-sum if and only if there exist $\alpha > 0$ and $\beta > 0$ such that $\alpha A + \beta B = Y + Z$ where Y is a matrix with identical columns and Z is a matrix with identical rows. Strategically zero-sum games are important because, for them, most of the desirable properties of zero-sum games still hold. However, since the split manufacturing game can not be guaranteed to be strategically zero-sum, we can not analyze it formally but empirically.

Future work. Recall that in this work, we assume that each design is sent to the foundry only once. Therefore, there is no notion of multiple rounds for each design and accordingly no scope of learning in between. However, in the future, we plan to investigate the setup of multiple rounds with learning enabled in between. The setting of multiple rounds would be as follows. The defender sends the layout to the foundry, and

TABLE VIII
KEY TAKEAWAY POINTS FOR EACH SCENARIO

Scenario	Takeaway points
Scenarios #1 and #2	(i) Irrespective of whether the attacker knows how likely the defender is to use any technique, the optimal strategies for the defender for OER and HD metrics are the same for both scenarios. (ii) For the OER metric, no defense technique is required for large benchmarks. (iii) It is not always optimal to use all hints in the network-flow attack; instead a subset of hints should be used.

the attacker attacks the FEOL layout and recovers some or all connections. Then, the designer would send another layout, but now, she/he has some additional information about how well his/her defense worked in the last round. In such a setting, the defender can learn the attacker's choice of attack technique and choose an appropriate defense technique or combination of defense techniques in successive rounds.

VII. Conclusion

Various attack and defense techniques have been proposed in split manufacturing; however, there has been no demonstration of a mathematical framework to evaluate the efficacy of the techniques. In this work, we took a different direction: instead of proposing yet another attack/defense technique, we developed a game-theoretic framework to find the optimal defense strategies. We evaluated our framework on five defense techniques with nine attack techniques for two security metrics, on ten diverse benchmarks for two attack scenarios. Following are the essential conclusions from our analysis.

1) Optimal strategies obtained using our framework help designers save unnecessary costs in terms of PPA overheads. For instance, consider the ITC-99 benchmark `b19_C`. According to our game-theoretic analysis, ND is the optimal defense technique for a designer when considering the OER metric. Therefore, the designer does not incur any PPA overheads that other defense techniques like RP or BP would have incurred.
2) We also uncover a counter-intuitive finding: using all the hints in the network-flow attack does not necessarily help the attacker. This observation highlights the need to develop more robust flavors of proximity attacks.
3) Based on our analysis, we observe that *the defender succeeds by selecting a subset of defense techniques without even knowing what hints the attacker will use a priori.* More importantly, the defender is not at a disadvantage if the attacker knows the probabilities with which the defender chooses her actions, i.e., scenario #1 does not lead to any additional loss for the defender than scenario #2.
4) Under our cost model and considered attacks, large-scale designs are inherently secure by naïve split manufacturing. This indicates that industrial designs (e.g., *IBM superblue*) might not require protection (either by placement and/or routing perturbation). Note that these observations are a result of our game-theoretic analysis.

Ramifications. Our observations have significant ramifications in the research of split manufacturing, particularly in the development of attack techniques. Since larger designs are secure by naïve split manufacturing, we urge the research community to redirect their efforts in developing stronger and scalable attacks. In fact, the OER for the larger benchmarks (after attack) is almost 100% even for the DL attack, which is potent than the network-flow attack. Hence, some foundational studies regarding the problem of split manufacturing are required along with the development of better (in terms of achieving higher CCR, ideally close to 100%) and scalable (in terms of being applicable on million-gate designs) attacks. Moreover, our game-theoretic framework would remain crucial and help the IC designers obtain the optimal defense technique and provide a cost-benefit trade-off even under the scenario when researchers develop a better attack strategy that thwarts the current state-of-the-art defense techniques.

Acknowledgments

We express our gratitude to Prof. Srinivas Shakkottai for discussions that helped shape the project. We thank Dr. Haocheng Li for helping with the deep learning attack and the anonymous reviewers for their comments.

References

[1] C. McCants. (2016) Trusted integrated chips (TIC) program. Last accessed the website on 10/27/2020. [Online]. Available: https://www.ndia.org/-/media/sites/ndia/meetings-and-events/divisions/systems-engineering/past-events/trusted-micro/2016-august/mccants-carl.ashx

[2] B. Hill, R. Karmazin, C. T. O. Otero, J. Tse, and R. Manohar, "A split-foundry asynchronous FPGA," *Custom Integrated Circuits Conference*, pp. 1–4, 2013.

[3] J. Rajendran, O. Sinanoglu, and R. Karri, "Is split manufacturing secure?" *Proc. Design, Automation and Test in Europe*, pp. 1259–1264, 2013.

[4] A. Sengupta, S. Patnaik, J. Knechtel, M. Ashraf, S. Garg, and O. Sinanoglu, "Rethinking split manufacturing: An information-theoretic approach with secure layout techniques," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2017, pp. 329–326.

[5] J. Magaña, D. Shi, J. Melchert, and A. Davoodi, "Are proximity attacks a threat to the security of split manufacturing of integrated circuits?" *Transactions on Very Large Scale Integration Systems*, vol. 25, no. 12, pp. 3406–3419, 2017.

[6] L. Feng, Y. Wang, J. Hu, W.-K. Mak, and J. Rajendran, "Making split fabrication synergistically secure and manufacturable," *International Conference On Computer Aided Design*, pp. 313–320, 2017.

[7] Y. Wang, P. Chen, J. Hu, G. Li, and J. Rajendran, "The cat and mouse in split manufacturing," *Transactions on Very Large Scale Integration Systems*, vol. 26, no. 5, pp. 805–817, 2018.

[8] S. Patnaik, M. Ashraf, J. Knechtel, and O. Sinanoglu, "Raise your game for split manufacturing: Restoring the true functionality through BEOL," in *Design Automation Conference*, 2018, pp. 140:1–140:6.

[9] H. Li *et al.*, "Attacking split manufacturing from a deep learning perspective," in *Design Automation Conference*, 2019, pp. 135:1–135:6.

[10] ——, "Deep learning analysis for split manufactured layouts with routing perturbation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020.

[11] S. Patnaik, M. Ashraf, H. Li, J. Knechtel, and O. Sinanoglu, "Concerted wire lifting: Enabling secure and cost-effective split manufacturing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[12] T. D. Perez and S. Pagliarini, "A survey on split manufacturing: Attacks, defenses, and challenges," *IEEE Access*, vol. 8, pp. 184 013–184 035, 2020.

[13] "ASIC Cost Effectiveness," http://smithsonianchips.si.edu/ice/cd/ASIC98/SECTION5.PDF, 1998, [Online; accessed 16-November-2020].

[14] L. A. Gordon and M. P. Loeb, "The Economics of Information Security Investment," *Transactions on Information and System Security*, vol. 5, no. 4, p. 438–457, 2002.

[15] "Intel Adds Core i9-10850K To Desktop Chip Lineup: 10 Cores Minus 100MHz," https://www.anandtech.com/show/15929/intel-adds-core-i910850k-to-desktop-chip-lineup, [Online; accessed 19-November-2020].

[16] B. Barry, "Software engineering economics," 1981.

[17] N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, "Algorithmic game theory, cambridge univ," 2007.

[18] J. F. Nash *et al.*, "Equilibrium points in n-person games," *Proceedings of the National Academy of Sciences*, vol. 36, no. 1, pp. 48–49, 1950.

[19] L. Amarú, P.-E. Gaillardon, and G. De Micheli, "The EPFL combinational benchmark suite," in *Proceedings of the 24th International Workshop on Logic & Synthesis*, 2015.

[20] "SplitExtract," https://github.com/cuhk-eda/split-extract, 2021, [Online; accessed 21-July-2021].

[21] "SplitAttack," https://github.com/cuhk-eda/split-attack, 2021, [Online; accessed 21-July-2021].

[22] "Average IC Designer Salary," https://www.payscale.com/research/US/Job=Integrated_Circuit_(IC)_Designer/Salary, [Online; accessed 16-November-2020].

[23] "2017 Government Contractor Survey," https://www.grantthornton.com/-/media/content-page-files/public-sector/pdfs/surveys/2018/2017-government-contractor-survey, 2018, [Online; accessed 16-November-2020].

[24] S. Khan and A. Mann, "AI Chips: What They Are and Why They Matter," *Center for Security and Emerging Technology*, 2020.

[25] "Early TSMC 5nm Test Chip Yields 80%, HVM Coming in H1 2020," https://www.anandtech.com/show/15219/early-tsmc-5nm-test-chip-yields-80-hvm-coming-in-h1-2020, 2019, [Online; accessed 22-July-2021].

[26] "Amazon EC2 On-Demand Pricing," https://aws.amazon.com/ec2/pricing/on-demand/, 2020, [Online; accessed 16-November-2020].

[27] J. Graf, "Trust games: How game theory can guide the development of hardware trojan detection methods," *International Symposium on Hardware Oriented Security and Trust*, pp. 91–96, 2016.

[28] J. Graf, W. Batchelor, S. Harper, R. Marlow, E. Carlisle, and P. Athanas, "A practical application of game theory to optimize selection of hardware Trojan detection strategies," *Journal of Hardware and Systems Security*, vol. 4, no. 2, pp. 98–119, 2020.

[29] C. A. Kamhoua, H. Zhao, M. Rodriguez, and K. A. Kwiat, "A game-theoretic approach for testing for hardware trojans," *Transactions on Multi-Scale Computing Systems*, vol. 2, no. 3, pp. 199–210, 2016.

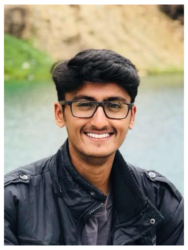
[30] A. M. Smith, J. R. Mayo, V. Kammler, R. C. Armstrong, and Y. Vorobeychik, "Using computational game theory to guide verification and security in hardware designs," *International Symposium on Hardware Oriented Security and Trust*, pp. 110–115, 2017.

[31] M. Breton, A. Alj, and A. Haurie, "Sequential Stackelberg equilibria in two-person games," *Journal of Optimization Theory and Applications*, vol. 59, no. 1, pp. 71–97, 1988.

[32] R. J. Aumann, "Correlated equilibrium as an expression of Bayesian rationality," *Econometrica: Journal of the Econometric Society*, pp. 1–18, 1987.

[33] A. R. Washburn *et al.*, "Two-person zero-sum games," 2014.

[34] H. Moulin and J.-P. Vial, "Strategically zero-sum games: the class of games whose completely mixed equilibria cannot be improved upon," *International Journal of Game Theory*, vol. 7, no. 3-4, pp. 201–221, 1978.



Vasudev Gohil received his B.Tech. degree in electrical engineering with a minor in computer science from the Indian Institute of Technology Gandhinagar, India in 2018. Shortly afterward, he joined Texas A&M University, College Station, TX, USA, where he is pursuing a doctorate in Computer Engineering. His research interests are at the intersection of hardware security, game theory, and optimization. In particular, he explores the application of machine learning and other practical and theoretical methods for improving the security of the IC supply chain.

Mark Tressler is an undergraduate student pursuing a B.S. in electrical engineering from Texas A&M University (TAMU), in College Station, TX, USA. He is currently working at TAMU as an undergraduate researcher. His current research interests include the use of new technologies to improve the field of hardware security.

Kevin Sipple is an undergraduate student pursuing a B.S. in electrical engineering and certification in engineering systems management from Texas A&M University in College Station, TX, USA. He currently works with at TAMU as an undergraduate researcher. His research interests reside in hardware security, power systems, and sustainability.

Satwik Patnaik received the B.E. degree in electronics and telecommunications from the University of Pune, India, the M.Tech. degree in computer science and engineering with a specialization in VLSI design from the Indian Institute of Information Technology and Management, Gwalior, India, and the Ph.D. degree in electrical engineering from Tandon School of Engineering, New York University, Brooklyn, NY, USA in September 2020.

He is currently a Postdoctoral researcher with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX, USA. His research delves into IP protection techniques, CAD frameworks for security, leveraging 3D paradigm for enhancing security, exploiting security properties of emerging devices, applied machine learning for hardware security, and side-channel evaluation.

Jeyavijayan (JV) Rajendran is an Assistant Professor in the Department of Electrical and Computer Engineering at the Texas A&M University. Previously, he was an Assistant Professor at UT Dallas between 2015 and 2017. He obtained his Ph.D. degree from New York University in August 2015. His research interests include hardware security and computer security. His research has won the NSF CAREER Award in 2017, the ACM SIGDA Outstanding Young Faculty Award in 2019, the ACM SIGDA Outstanding Ph.D. Dissertation Award in 2017, and the Alexander Hessel Award for the Best Ph.D. Dissertation in the Electrical and Computer Engineering Department at NYU in 2016, along with several best student paper awards.