Physical Design Obfuscation of Hardware: A Comprehensive Investigation of Device- and Logic-Level Techniques

Arunkumar Vijayakumar, *Student Member, IEEE*, Vinay C. Patil, *Student Member, IEEE*, Daniel E. Holcomb, *Member, IEEE*, Christof Paar, *Fellow, IEEE*, and Sandip Kundu, *Fellow, IEEE*

***Abstract*— The threat of hardware reverse engineering is a growing concern for a large number of applications. A main defense strategy against reverse engineering is hardware obfuscation. In this paper, we investigate physical obfuscation techniques, which perform alterations of circuit elements that are difficult or impossible for an adversary to observe. The examples of such stealthy manipulations are changes in the doping concentrations or dielectric manipulations. An attacker will, thus, extract a netlist, which does not correspond to the logic function of the device-under-attack. This approach of camouflaging has garnered recent attention in the literature. In this paper, we expound on this promising direction to conduct a systematic end-to-end study of the VLSI design process to find multiple ways to obfuscate a circuit for hardware security. This paper makes three major contributions. First, we provide a categorization of the available physical obfuscation techniques as it pertains to various design stages. There is a large and multidimensional design space for introducing obfuscated elements and mechanisms, and the proposed taxonomy is helpful for a systematic treatment. Second, we provide a review of the methods that have been proposed or in use. Third, we present recent and new device and logic-level techniques for design obfuscation. For each technique considered, we discuss feasibility of the approach and assess likelihood of its detection. Then we turn our focus to open research questions, and conclude with suggestions for future research directions.**

***Index Terms*— Logic obfuscation, circuit obfuscation, reverse engineering, device manipulation, circuit manipulation.**

I. Introduction

THE threat of hardware reverse engineering is a growing concern for a large number of applications, from consumer electronics and cyber-physical systems all the way to military systems. The are two broad goals an adversary might pursue: (1) learning the functionality of the device-under-attack and (2) gaining knowledge that enables manipulation of the target. With respect to the first goal, reverse engineering of ICs may be motivated by a number of reasons. For example, the adversary may wish to steal IP for their own use, or to resell IP without licensing. They may also wish to analyze a device to gain a competitive advantage or to leapfrog technology. Regarding the second goal, active design manipulations, a particularly attractive target for attackers are implementations of hardware-based security functions such as cryptographic algorithms or random number generators. An attacker may want to learn the details of a security module with the goal of subsequently weakening it or inserting carefully crafted Trojans. In addition to manipulating security modules, an adversary can also be interested in adding or disabling functionality to commercial ICs [1]. Note, however, that a reverse engineer does not necessarily need to pursue illegal or ethically questionable objectives. An "adversary" in the reverse engineering setting can also represent a stakeholder attempting to discover IP violations or faulty product designs. Furthermore, reverse engineering for competitive reasons is legal in most Western countries, including the United States and the European Union [2]. A further discussion of adversary goals is given in Section II.

The problem of hardware reverse engineering is not only theoretical or limited to powerful adversaries. Numerous hardware attacks over the last few years have shown the ease of removing coating and de-layering circuits [3]–[7]. Moreover reverse engineering is also increasingly supported by CAD tools such as Chipworks' ICWorks and the open-source tool Degate [8]. In addition, reverse engineering equipment such as scanning electronic microscopes and Focused Ion Beam (FIB) have become considerably more accessible over the past decade due to wide-spread availability in research labs and a drop in price for new and used equipment.

An IC designer who wants to prevent reverse engineering has a difficult task because the adversary will have physical possession of the chip. Since physical anti-tamper measures to prevent invasive attacks are technically and economically infeasible in most applications, the designer's main line of defense is obfuscation of the hardware. There are two principal

Manuscript received March 3, 2016; revised June 19, 2016; accepted August 1, 2016. Date of publication August 17, 2016; date of current version October 31, 2016. This work was supported by NSF under Grants 1421352 and 1563829. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Ozgur Sinanoglu.

A. Vijayakumar, V. C. Patil, D. E. Holcomb, and S. Kundu are with the Department of Electrical and Computer Engineering, University of Massachusetts Amherst, Amherst, MA 01003 USA (e-mail: avijayak@umass.edu; vcpatil@umass.edu; dholcomb@umass.edu; kundu@umass.edu).

C. Paar is with the Department of Electrical and Computer Engineering, University of Massachusetts Amherst, Amherst, MA 01003 USA, and also with the Horst Görtz Institute for IT-Security, Ruhr-Universität Bochum, 44801 Bochum, Germany (e-mail: christof.paar@rub.de).

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/TIFS.2016.2601067

approaches to hardware obfuscation: *structural obfuscation* and what we coin *physical design obfuscation*.[1]

Structural obfuscation — which tries to hide the true functionality of the device-under-attack through techniques such as randomized placement of logic elements, irregular routing or dummy wires — is widely used in industry. Unfortunately, structural obfuscation does not prevent the adversary from recovering the complete netlist through de-layering. With structural obfuscation, she faces the task of learning the functionality from the recovered netlist which has a structure that makes this very task more complicated. However, given enough resources and time, an attacker will most likely succeed. This situation is analogous to software reverse engineering. In both cases, the designer does not have a true advantage over the attacker, i.e., the situation is symmetric in the sense that the adversary can observe whatever methods the designer employs.

This paper addresses the second family of techniques, physical design obfuscation, which performs alterations of circuit elements which are *difficult or impossible to observe* for the adversary. Examples of such stealthy manipulations are localized changes in doping concentrations, dielectric manipulation, and open connections in the sub-nanometer range. In contrast to "classical" structural obfuscation, when stealthy manipulations are used, the adversary will extract a netlist which does *not* unambiguously correspond to the logic function of the device under analysis (DUA). Gate camouflaging is one such approach to physical design obfuscation [9], [10]. Gate camouflaging and similar techniques are promising, as they give the designer a true advantage of knowledge over the adversary. For a successful reverse engineering attack, the adversary has to recover both the basic design structure and also the stealthy manipulations; the former is easy to recover while the latter should be hard. This situation is akin to classical cryptography in which an cryptographic algorithm is known while the key is secret and must be learned by the attacker. Due to the asymmetry of the setting which gives the designer an advantage, this approach to obfuscation has the potential to provide strong protection.

Despite the promise of this approach, there is scant treatment in the literature and industry on this topic. The contribution at hand attempts to give a comprehensive treatment to the emerging and promising area of physical design obfuscation. The main contributions of this paper are as follows:

- We present a taxonomy of physical design obfuscation techniques (Sec. III).
- We investigate existing and novel physical design obfuscation techniques at the device level (Sec. V)
- We investigate novel methods for employing physical design obfuscation at the logic-level (Sec. VI) by utilizing the device-level techniques.

To the best of our knowledge, this is the first time that the topic of physical design manipulations for obfuscation is taxonomically surveyed. We believe this work will be valuable for industry, where ad-hoc methods dominate the issue of circuit protection through obfuscation. The proposed work also has the potential to initiate further research related to physical design mechanisms, as well as research in realizing complex circuits and evaluating them with respect to resistance to reverse engineering.

The paper is organized as follows. Section II discusses the adversary setting and Section III introduces a taxonomy for physical design obfuscation. In Section IV we summarize related work. As main contributions, we discuss in Sections V and VI the various physical manipulations and logic-level mechanisms, respectively, that can be used for obfuscation. The paper concludes with a discussion of open challenges in Section VII.

II. Adversary Setting

Hardware obfuscation is a tactic used by a designer to hide functionality from an unauthorized third party. In order to systematically treat the somewhat murky question of hiding, two aspects about adversaries need to be addressed. First, what are the possible objectives of an attacker, and second, what are her capabilities? The answers to these questions influence how successful a given obfuscation technique can be.

We note that the chip manufacturer (foundry) has perfect information about the layout on all levels, including the details about physical obfuscation methods. Hence, a manufacturer can, at least in theory, understand circuit functionality despite all physical obfuscation methods that may be employed. However, in most civilian applications, the designer's primary concern is not an adversary who is in cohort with a foundry. The far more common case is an adversary who is in possession of one or more obfuscated ICs, and who attempts to extract information about the circuit. As we will see below, attackers can vary widely with respect to goals (which information he wants to extract) and his capabilities (what he can observe and manipulate). This class of attack is sometimes referred to as Man-at-the-End (MATE) attack in software security community [11].

A. Objectives of the Adversary

The most obvious goal of an attacker — to extract the complete function of the circuit realized by the device-under-attack from its physical structure —- is only one of many. For instance, in competitive analysis, the adversary might merely be interested in learning which algorithms are being used in a given product. Such objectives are often less complex and, thus, easier to achieve. We refer interested readers to the work of Rostami *et al.* [12] for more background on this issue. As noted earlier, a hardware reverse engineer does not necessarily need to pursue illegal or ethically questionable objectives, but she can also attempt to discover IP violations or faulty product designs. Below we list some common goals of an adversary, roughly ordered from easiest to hardest.

1) Learning the device properties and design rules of fabricated ICs (process analysis). For instance, the estimated design rules for an Intel 22nm tri-gate process can be obtained by commercially-offered reverse engineering for $16,000 USD [13].

[1]We would like to clarify the terminologies: The terms physical design/physical mechanism used in this paper refer to device and interconnect-level techniques that can be used for obfuscation. It does not refer to physical design methods used for placement and routing of circuits.

2) Learning the location of sensitive signal wires or buses. This information can enable subsequent use of targeted micro-probing to extract internal data values. These values may be immediately relevant as in the case of cryptographic keys, or the values may provide more information about the design, e.g., state bits of an Finite-state machine (FSM) or register values.
3) Learning which specific IP blocks or algorithms are being used in the target design. This does not require learning the details of the implementation. Possible motivations include competitive intelligence or detection of IP violations.
4) Learning the Boolean function of combinational logic, or the FSM of sequential logic (circuit extraction). This can apply to the entire IC or only certain parts which are proprietary and/or sensitive. The adversary's goal might be to clone or even alter the product.
5) Learning the logic function of each gate and the interconnections between them. This implies learning the overall Boolean function, but also includes additional information about exactly how the function is implemented, and the extra information could enable side channel attacks etc. The incentives for this are the same as in the previous point, and an additional incentive is for performing detailed competitive analysis.
6) Learning the GDSII of an entire IC or of parts, e.g., an IP block. This will enable the adversary to re-manufacture an identical IC in the same technology. This is the highest value information that can be obtained by reverse engineering.

B. Capabilities of the Attacker

The effectiveness of obfuscation will depend on the assumed capabilities of the attacker. Publicly available tools are available to facilitate these methods of attack [8]. Torrance and James [2] give an overview of the state of the art for reverse engineering in 2011. The following provides some attacker capabilities ordered roughly from weakest to strongest.

1) The attacker is able to observe all I/O activities and can create and manipulate input values at will.
2) The attacker is able to decapsulate the IC and to probe values on block-level I/O boundaries or top-level metal layers between blocks with approximately 1k–10k gates.
3) The attacker is able to observe the state of sequential elements at any time as would occur with scan chain access; Rajendran *et al.* [9] assume their attacker to have this ability.
4) The attacker is able to delayer chip and learn all metal layers and corresponding vias but is not able to distinguish real from dummy vias; this implies ability to identify gate structures as done by Nohl *et al.* [3] and others.
5) The attacker is able to delayer chip and resolve sub-gate-level width and spacing of metal or polysilicon features of layers. Torrance and James [2] note that optical imaging suffices to extract such finer details at technologies above 0.18μm, and that Scanning Electron Microscope (SEM) can accomplish the same below 0.18μm if the dielectric is removed.
6) The attacker is able to detect dopant programming of transistors using techniques like passive voltage contrast [14].
7) The attacker is able to delayer chip and further can distinguish between vias and fake vias; Rajendran *et al.* [9] and others assume an attacker that lacks this capability.

A summary of various measurement techniques that can be used in reverse engineering is presented in Table I.

TABLE I
VARIOUS MEASUREMENT TECHNIQUES USED IN REVERSE ENGINEERING

Technique	Capability	Cost
PICA imaging [15]	Observe transistors switching	High
SQUID Microscopy [16]	Observe constant current from shorts	High
Scanning Electron Microscopy (SEM)	Imaging of features below 200nm [2]	Moderate
Optical Microscopy (OP)	Imaging of features down to 200nm [2]	Low
Passive Voltage Contrast (PVC)	Detect Doping	Low

III. A TAXONOMY FOR PHYSICAL DESIGN OBFUSCATION

In the literature, low-level hardware obfuscation has rarely been treated in a systematic way. At the same time, there is a large and multi-dimensional design space for introducing hardware obfuscation. In fact, there is a wealth of physical design obfuscation techniques which result in stealthy circuits, i.e., circuits whose realized logic behavior differs from the apparent logic function extracted during reverse engineering. We propose a taxonomy in order to systematically treat this multifaceted topic. In fact, many obfuscation techniques often blur the boundaries between these layers. Nevertheless, these layers are helpful for systematic organization of apparently disparate physical manipulation techniques. In normal VLSI design process, successive transformations from one abstraction layer to the next result in the final physical design. However in obfuscation, the goal is to add/subtract some information specific to cross-layer transformations. Therefore, describing obfuscation purely in terms of extracting the higher levels of abstraction from lower level is complicated and may not always be possible. For example, we can introduce layout artifacts to deliberately induce obfuscation through signal integrity which has no gate-level functional equivalent at the logic abstraction layer. It is for this reason, we introduce a different nomenclature for expressing obfuscation instead of using standard VLSI design abstractions.

Hardware obfuscation based on stealthy circuit manipulations can be viewed in a three-layer model, shown in Fig. 1. The lower two layers are formed by (i) device and interconnect mechanisms (bottom layer) and logic-level mechanisms (middle) layer. They are in hierarchical structure, i.e., logic-level mechanisms are based on the device and interconnect manipulations. These two layers are forms of physical design obfuscation and are the subject of this contribution. Physical design obfuscation provide the basic mechanisms with which actual obfuscation techniques can be realized. Such techniques form the top layer in Fig. 1. We elaborate on each of the three layers below.

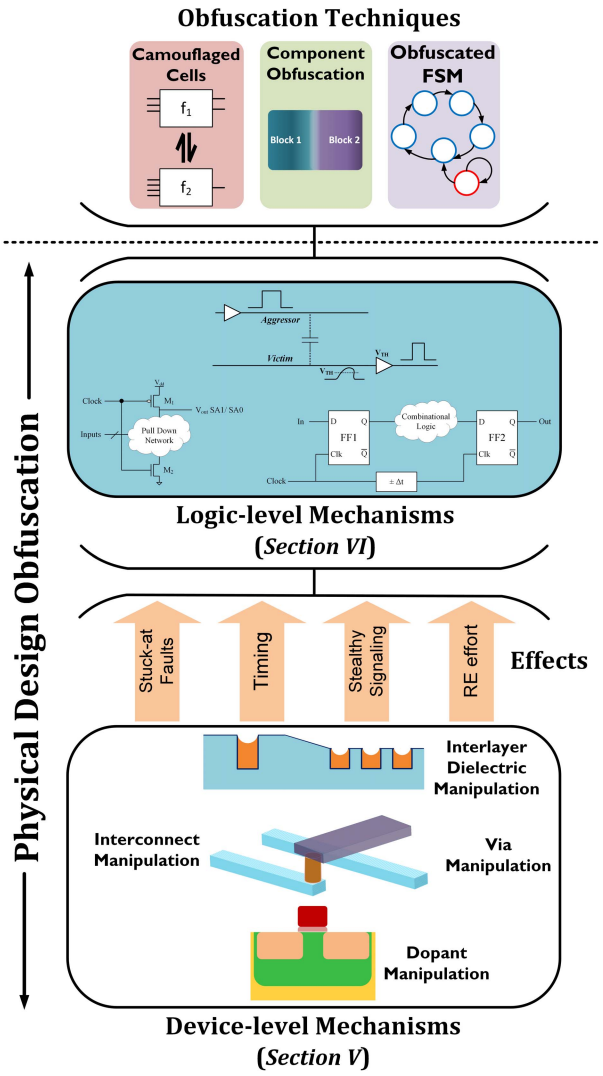

Fig. 1. A structured view on physical design obfuscation: device & interconnect mechanisms on the bottom layer and logic mechanisms on the middle layer enable actual obfuscation techniques on the top layer.

Bottom Layer: Device-level mechanisms refer to measures such as re-sizing of transistors or interconnect manipulations. Even though there are numerous such possible alterations, they can be classified into three groups. Each of the groups has a certain type of *effect* on the circuit, as shown in Fig. 1: (*i*) stuck-at-faults, (*ii*) stealthy signaling and (*iii*) delay manipulation.

Stuck-at-faults refer to changes that cause certain circuit nodes to have a constant logical 0 or 1, even though the apparent circuit structure might imply some other function.

Stealthy signaling refers to situations where an adversary may be unable to tell whether physical structures are communicating, and this includes both direct connection and non-contact influencing mechanisms such as cross-talk.

Delay manipulation refers to techniques that make a node switch slower or faster than it would appear to, and uses this to change the sequential behavior of a chip.

Middle Layer: Logic-level mechanisms The logic level is concerned with circuit structures which are built from the stealthy manipulations from the bottom layer described above. In subsequent higher system layers, the logic-level elements are used as building blocks for actual obfuscation techniques to protect integrated circuits. These logic-level mechanisms form the second level of abstraction and can to some extent be independent of the specifics of the device-level mechanisms (i.e., sizing, dopant, etc). For example, one can design a circuit that uses physically obfuscated look-up tables while abstracting away the lower level, i.e., what physical mechanisms are actually used to create the obfuscated look-up tables.

Top Layer: Obfuscation Techniques The last layer of the obfuscation hierarchy is used by techniques such as gate camouflaging, component obfuscation and FSM obfuscation. These higher-layer techniques can be used to protect complex circuits and systems that are realized on an IC. There have been several interesting proposals for securing hardware using obfuscation techniques.

Most of the higher-level obfuscation techniques on the top layer of Fig. 1 have been published relatively recently, cf. Section IV. However, there is a void regarding which underlying physical design mechanisms exist for realizing higher-level obfuscation. The focus of the contribution at hand is to provide a comprehensive treatment of the lower-level mechanisms. First, we introduce in a structured way a variety of device and interconnect mechanisms which can be used for physical design obfuscation in Section V. The second focus is on logic-level mechanisms which are discussed in Section VI, i.e., techniques which belong to the middle layer in the figure.

IV. Related Work

Even though hardware obfuscation has been used for a long time in commercial and military ICs, the topic had received scant treatment in the scientific community. However, over the last few years, the general area of hardware protection has become more active. Roughly speaking, the proposed methods can be classified into system- and circuit-level techniques, and we summarize related work below.

A. System-Level Techniques

Some of the system-level methods that have been proposed can be viewed as "classical" obfuscation approaches, whereas others are more general techniques for circuit protection, i.e., their focus is not directly on hiding of elements.

1) Component Obfuscation: Many reverse engineering techniques engage in *component recognition*. Techniques have been proposed to increase the complexity of delineating components by changing the logic gates at the output and input boundaries of connected blocks (Boundary Blurring) [17], replacing an entire circuit with an obfuscated equivalent (Component Fusion) [18], [19] and targeting interconnections between two blocks for encryption [20].

2) Obfuscation With Programmable Logic: Wendt *et al.* propose replacing subset of gates of a circuit with Physically Unclonable Function (PUF) and FPGA-based logic [21].

The PUF is assumed to be unique and unclonable, and the connected FPGA is used to reproduce functionality of the replaced logic. This approach provides a custom solution for each chip but it comes at considerable area and run-time costs.

3) FSM Modification: Alkabani *et al* [22], [23] propose FSM modification techniques for the purpose of locking a chip by obfuscating its power-up state. Only the correct key input sequence allows the circuit to be initialized correctly. Li and Zhou [24] propose a methodology for obfuscating sequential circuits and embedding a secret key in the power-up state of the IC that must be present to unlock full unthrottled performance. Chakraborty and Bhunia [25] propose a methodology to perform simultaneous obfuscation and authentication of an SoC design netlist.

4) Related Techniques — Logic Encryption: Logic encryption is a relatively recent hardware security concept introduced by Roy *et al.* [26]. In logic encryption, additional gates are added to the design such that only the correct input values, considered the key, allow the circuit to work as intended. Wrong values can lead to corrupted output [27] or even can lock the circuit. Logic encryption can be an effective tool against over production, however, it is not effective against invasive attacks. It has been shown that it is possible to reverse engineer logic encryption keys in time that is linear in the number of keys [28] and quite efficient in practice using Boolean satisfiability problem (SAT) solving [29]. An attacker may also be able to observe the added logic and trace the key inputs by using imaging techniques.

B. Circuit-Level Obfuscation

The primary goals of circuit-level obfuscation methods have been to create or modify cell libraries to hide gate functions and adding non-essential structures to the design. The underlying assumption for these approaches is that an adversary is presented with extra complexity when reverse-engineering the logic.

1) Camouflaged Cells: Obfuscation via camouflaging of cells can be achieved through custom design to either mimic other cells or to allow for post-manufacturing programmability to customize gate functions. Relevant literature dealing with cell camouflaging can be found in patents by Baukus *et al.* [30], [31] and Cocchi *et al.* [32].

2) Filler Cells: Chow *et al.* [33] and Cocchi *et al.* [10] deal with utilizing filler cells with routing in a realistic fashion to create a dense network. Some of the cells may even be connected to the functional logic but do not hamper its operation. The filler cells and their associated routing increase the amount of data that an attacker will need to sort through during reverse engineering in order to extract the underlying logic.

C. Hardware Security Surveys

Erstwhile research has resulted in extensive surveys on protecting hardware intellectual property (IP). The work by Tehranipoor and Wang [34] provides a good understanding of various aspects of hardware security and trust with focus on all types of electronic devices and systems. Colombier and Bossuet [35] provide a detailed discussion on protection of design data and IP with exhaustive survey of previous research into hardware protection. Guin *et al.* [36] provide a thorough analysis of the problem of counterfeiting and examine various countermeasures including hardware obfuscation. These papers provide a summary of various hardware obfuscation techniques proposed in literature. In contrast, the focus of this paper is on surveying *physical design manipulation mechanisms* and presenting device- and logic-level techniques for hardware obfuscation.

V. Device-Level Mechanisms

In this section, we discuss physical design obfuscation through device-level techniques, which we consider to be manipulations of transistors and the interconnections between them. These obfuscations variously aim at creating stuck-at faults, delay faults, virtual connections which are stealthy to the attacker, or to increase the reverse engineering effort by introducing extraneous information. A summary of the impacts of these device-level techniques is given at the end of this section in Tab. II. In the following we focus on the most significant ways for each mechanism to influence the behavior of the design, but note that there is significant overlap and ambiguity between them, for example, steathly signaling, and stuck-at faults or timing faults.

A. Device Specific Mechanisms

1) Source/Drain Doping for Stuck-at or Timing Faults: In semiconductor manufacturing doping is one of the major techniques to modify the electrical characteristics of transistors. Doping concentration can be used to change transistor characteristics with little or no change in transistor geometry, which makes it hard for an adversary to observe them directly. Fig. 2 shows the layout of a PMOS transistor; under normal conditions, the source and drain regions are doped with P-type dopant. If those regions are instead doped with N-type dopant, a short circuit is created between the drain and source terminals. This type of dopant manipulation has been used to create stuck-at faults and create Trojans [37]. Similarly, a circuit with such dopant characteristics can be used to confuse the attacker by providing pretentious information about the circuit's functionality. An example of how this can be exploited for obfuscating CMOS circuit is discussed in Section VI-A1.

2) Channel Doping for Stuck-at or Timing Faults: Similar to source/drain doping, channel doping can be varied to change the characteristics of transistor. By controlling channel doping, a transistor can be configured as depletion or enhancement type. In an enhancement type, the channel is implanted rather than induced, the transistor is always `on`. This can be used to create stuck-at faults. An example of how this can be used for obfuscating CMOS circuit is discussed in Section VI-A1.

Multi-threshold design is a well-known technique where multiple threshold transistors are used in the same digital design to achieve power-performance trade-offs [38].

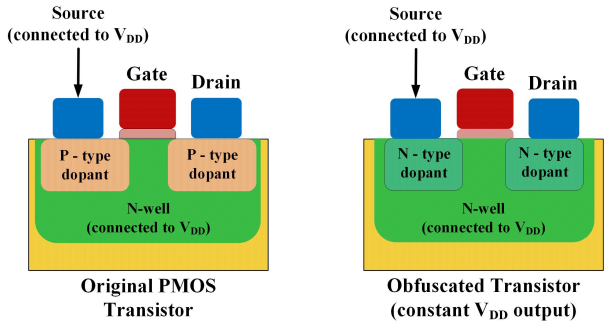

Fig. 2. Use of atypical doping to make an apparent PMOS transistor realize a constant-V_{DD} output.

As the threshold of a transistor depends on channel doping, channel doping can be used to subtly manipulate performance to create non-obvious designs. Examples of how threshold voltage can be exploited in flip-flops, pass transistors and dynamic logic circuits are discussed in section Sections VI-A4, VI-A2 and VI-A3, respectively. Multiple publications and patents deal with the usage of dopant manipulation techniques to achieve obfuscation [30]–[32], [39]–[41].

3) Reversing Dopant Obfuscation: From an attacker's perspective, even though doping does not change geometry, it is possible to detect source/drain doping changes by using passive voltage contrast (PVC) method [14]. However, PVC requires the use of scanning electron microscope (SEM) or focused ion beam (FIB), and this makes PVC analysis slow, especially when there are millions of transistors involved. An attacker can also use picosecond imaging circuit analysis (PICA) [15] to detect channel doping, but this is more expensive than PVC [14].

B. Interconnect Specific Mechanisms

1) Manipulating Inter-Layer Dielectric for Timing and Stuck-at Faults: Chemical Mechanical Planarization (CMP) is the process used to flatten each fabricated layer of a circuit after printing the layer features. CMP can adversely affect the final layer profile due to the phenomena of *dishing* and *erosion* as shown in Fig. 3 [42]. The extent of dishing and erosion depends on line width and density [42]. Metal-fills help to provide line density uniformity across chip to allow for predictable CMP performance, and for this reason pattern density is commonly constrained by design rules. A basic model for the relation of ILD thickness (z) and pattern density at a given location (x, y) was proposed by Ouma [43] and is given by (1). Manipulating the variables of (1) allow deliberate control of dishing and erosion for the purpose of obfuscation. For example, metal-fills surrounding an interconnect line can degrade the line during fabrication if placed at improper distances from it [44]; in obfuscation scenarios, this can be used to create a deliberate stuck-at or timing fault.

$$z = \begin{cases} z_0 - \dfrac{Kt}{\rho_0(x,y)} & \text{if } t < \dfrac{\rho_0 z_1}{K} \\ z_0 - z_1 - Kt + \rho_0 z_1 & \text{if } t > \dfrac{\rho_0 z_1}{K} \end{cases} \quad (1)$$

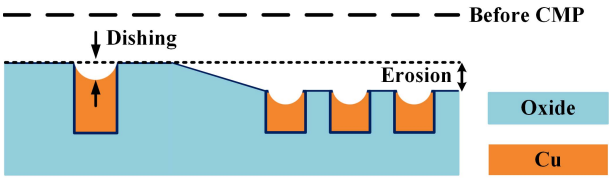

Fig. 3. Dishing and Erosion.

where:

- K oxide polishing rate
- z_0 thickness of oxide deposition
- z_1 initial step height
- t total polish time
- $\rho_0(x, y)$ initial oxide pattern density before CMP

2) Dummy Logic to Increase Reverse Engineering Effort: In semi-custom design flows, with cell rows and standard cells, not all rows can be fully utilized by the design. The resultant unused space is filled with extra gates, called back-fills. This extra logic is useful to make late-stage design changes, for debug, or for replacing bad cells during fabrication. Obfuscation can be achieved by combining unused back-fills with metal-fills to create junk logic. This increases complexity of the design that an attacker needs to decipher. Effective utilization of metal fills to harden a design against an attacker is proposed by Baukus *et al.* [45] and Cocchi *et al.* [10]. In addition to increasing the volume of data that an attacker needs to process, there is also increased uncertainty as to which metal traces are real and which are extraneous. Further, connecting the metal fills to power, ground, switching signals or keeping them floating reduces the effectiveness of reverse engineering techniques like voltage contrast.

3) Stealthy Signaling Using Crosstalk: Metal-fill can also be used for stealthy signaling by exploiting the parasitic capacitance between neighboring interconnects [46], [47]. Metal-fills are usually connected to VDD/ GND to reduce crosstalk effect on nearby signal lines. If the Metal-fills nearby to a target signal line are connected to a clock or other controllable line, then they will induce strong and controllable crosstalk on the signal line. For example, using crosstalk in this manner, the Metal-fill can raise an interrupt signal by inducing a noisy signal higher than the threshold level (V_{TH}) of the buffer on an interrupt line. Through proper design, only the designer has knowledge of when the interrupt will be raised. This can be utilized to obfuscate the functioning of a particular block by creating a secret interrupt unobservable by an attacker. We illustrate this in Fig. 4.

Manipulation of Inter-Layer Dielectric (ILD) is a potential control knob for facilitating stealthy capacitive signaling using crosstalk. ILD is made from low-κ dielectric materials to enhance interconnect electrical performance by reducing capacitive coupling and the resultant crosstalk noise between adjacent lines. Air pockets are also introduced to further reduce the effective κ but care should be taken not to compromise the mechanical strength of the ILD [48]. To support obfuscation, a particular ILD layer can be thinner than normal or made of higher κ material to increase capacitive coupling between

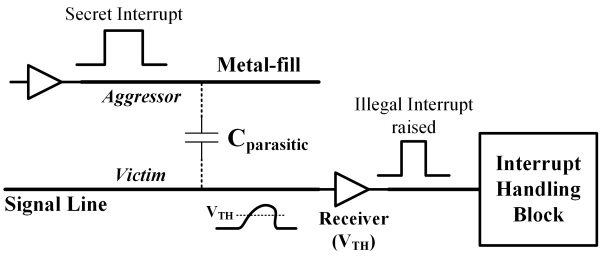

Fig. 4. Illustration of a Metal-fill influencing a signal line via capacitive coupling.

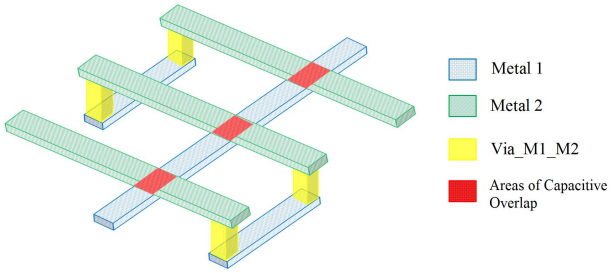

Fig. 5. Zigzag routing for crosstalk.

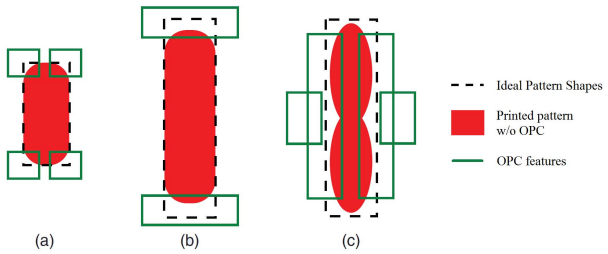

Fig. 6. OPC compensation techniques: (a) Serifs for Corner-rounding, (b) Hammerheads for Line-end shortening and (c) Line Biasing for Line-width shrinking.

interconnects across adjacent layers and increase crosstalk. Crosstalk model for two adjacent wires is shown in (2) and (3). Since metal lines in adjacent layers run perpendicular to each other, one can maximize crosstalk by running one interconnect, the *victim*, in one layer and another interconnect, the *aggressor*, in adjacent layers in a zigzag pattern (Fig. 5). ILD thinning maximizes the crosstalk induced by this inter-layer zig-zag pattern. Without extensive electrical measurement to estimate the magnitude of crosstalk that may occur between a pair of neighboring lines, a reverse engineer will find it difficult to extract any functionality that depends on crosstalk.

$$\Delta V_{victim} = \frac{C_{adj}}{C_{gnd} - v + C_{adj}} \frac{1}{1+k} \Delta V_{aggressor} \tag{2}$$

$$k = \frac{\tau_{aggressor}}{\tau_{victim}} = \frac{R_{aggressor}\left(C_{gnd-a} + C_{adj}\right)}{R_{victim}\left(C_{gnd-v} + C_{adj}\right)} \tag{3}$$

4) Via Alterations for Stealthy Signaling: Via manipulation techniques for obfuscation rely on the fact that most reverse engineering efforts delayer the chip without detailed inspection of the vias that connect the layers to each other. An attacker usually assumes the existence of a via based on the information from the metal layers, thereby recovering incorrect logic. Chow *et al.* [49] first proposed using a dummy via to give the appearance of a connection between a M1 layer metal and source/drain of a transistor. However, the inserted via ended in a field oxide close to the active layers of the transistor. Chow *et al.* [50] later proposed using dummy vias between interconnects on two metal layers such that there appears to be a connection, but in reality there is not. Rajendran *et al.* [9] have showcased the use of a generic layout with multiple via sites. The vias are either designated as true or dummy during design, and the same layout is used to implement multiple functionality.

The process of via formation has been described extensively [51], [52]. Controlling via formation provides another venue for obfuscation. One possible obfuscation technique is to increase ILD thickness such that the via is not fully formed or is very thin and breaks during burn-in. This can be utilized to create stuck-at faults in a circuit known only to the designer. Effective ILD thickness can be increased locally without affecting the rest of the layer as explained previously by using metal-fill density to control *dishing* and *erosion* as shown in Fig. 3 [42].

5) Lithographic Printability Features to Manipulate Timing: Optical Proximity Correction (OPC) and Sub-Resolution Assist Features (SRAFs) are techniques used to improve the printability of patterns on a fabricated chip. In obfuscation, these techniques can be used to give fine-grained control over fabricated structures and their delays. OPC addresses three major types of pattern distortions that occur during the fabrication process — corner rounding, line-end shortening and line-shrinking. These are compensated for by use of *serifs*, *hammerheads* and *line-biasing* OPCs, respectively [53], [54]. The various distortions and their associated compensations are illustrated in Fig. 6 [54].

In obfuscation, changing or removing *line-biasing* compensation enables thinning of the interconnect and increasing the line resistance. This affects the delay through the interconnect due to changes in the *RC* time constant, where R is the line resistance and C is the capacitive load on the line. The delay change can be utilized by the designer to introduce delay faults between flip-flops where a particular signal is biased such that it always arrives late and is never latched to the next stage. If a reverse engineer is unaware of this, they would infer incorrect logic from this line.

Manipulation of *hammerheads* can similarly slow down signals when they cross layers through vias at the end of the lines. Size of the *hammerheads* account for the effects of lithography, etching and also, for mask misalignment tolerances. By shaping or removing the *hammerheads* and changing the line alignments we can create an open on the interconnect crossing layers. Also, we can create a weak connection between two lines on different layers by partial misalignment, as illustrated in Fig. 7 where Δx and Δy alignments control whether the lines are connected or not.

SRAFs are features smaller than the smallest required feature printed on a die. They are used to improve the printability of the desired patterns during the lithography process, equalize lithographic performance between isolated and densely placed

TABLE II
SUMMARY OF FAULT MECHANISMS AND CORRESPONDING OBFUSCATION CLASSES AND DETECTION SCHEMES

			Obfuscation Classes			Detection
			Stuck-at Faults	Timing Faults	Stealthy Signaling	
Physical Design Mechanisms	Doping	Source/Drain	✗			PVC
		Channel	✗	✗		PICA
	Metal-fill		✗	✗	✗	SEM
	ILD Manip	Thinning		✗	✗	-
		Thickening	✗	✗	✗	-
	Intercon mask manipulation		✗	✗	✗	SEM, SQUID
	SRAFs			✗		-

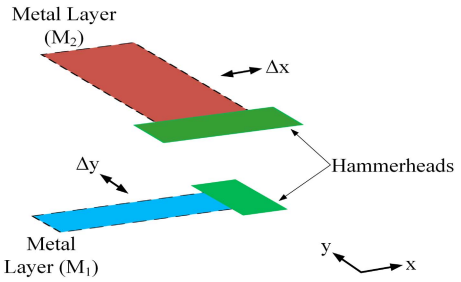

Fig. 7. Interconnect mask lines with Hammerheads in two adjacent layers.

features, and increase wafer yield. Melvin *et al.* [55] explain where SRAFs are usually placed. Many works [56]–[58] also deal with methodology for efficient placement, shaping and sizing of SRAFs. While proper SRAF placement improves printability of the target feature, primarily, via reduction in edge placement error [56], it is also possible to degrade the feature by altering the placement [58]. Thinning the line using SRAFs can be utilized in the same manner as *line-biasing* OPC to change resistance and cause timing faults. Ideally SRAFs should not be printed. In reality, there are tiny artifacts that are left behind due to SRAFs after completion of the manufacturing process. It is possible to size the SRAFs such that these final artifacts are of significant size to affect the legal features adjacent to them, like adding parasitic capacitance to a nearby interconnect.

OPC and SRAF-based manipulations will require high resource investment to implement. Many factors that might affect the printing of features during manufacturing are difficult to control. SRAFs increase the number of constraints in the design rules and their properties are adjusted constantly during post-layout lithographic simulations to improve yield. Hence, a designer will need to work with the foundry engineers to effectively utilize OPC and SRAFs for obfuscation. However, the high expense may have a corresponding high payoff, as the small manipulations caused by OPC and SRAFs will also be very difficult for an attacker to account for in his models.

6) Reversing of Interconnect Manipulations: For the techniques that use mask manipulations or metal-fills, an attacker using imaging techniques, like SEM, can only observe features in two dimensions (x,y). The delayering process removes the vertical dimension information. The attacker will require SEM to study the interconnect thickness variations to comment on changes in time constants. The interconnect obfuscation mechanisms are more useful in the lower metal layers where the majority of the signals are routed and the interconnect density is high. An attacker will have to invest a large amount of time to find potential sites of interest.

Also, interconnect misalignment techniques may not be detected due to the requirement of correlating measurements in two different lower metal layers which may fall below the resolution of the microscopy technique used. It is possible for an attacker to detect open/short faults on interconnects of interest using Superconducting Quantum Interference Device (SQUID) microscopy [16]. The attacker will still need to narrow the search to the target area by using other electrical techniques. To get information about the dielectric thickness of various layers, an attacker will need to slice the chip and observe the vertical layers through some microscopy technique. However, the dishing and erosion effects are localized and may not be reflected at the plane of the cut. Hence, inter-layer dielectric manipulation to affect crosstalk and via alterations cannot be easily found through microscopy techniques.

VI. LOGIC-LEVEL MECHANISMS

The second aspect of physical design obfuscation is concerned with logic-level structures. Circuit structures are needed to translate the low-level device and interconnect obfuscation mechanisms of the previous section into building blocks, which can be used for realizing complex sequential logic functions. These (primarily combinational) circuit structures form the next level of abstraction. As mentioned earlier, they can sometimes be agnostic of the underlying physical mechanisms (i.e. sizing, dopant, etc). In this section, we discuss novel logic-level techniques for obfuscation based on device-level mechanisms described in Section V.

A. Circuit Transformations for Obfuscation

The device-level mechanisms described in Section V have to be coupled with circuit topologies to create obfuscated circuits. The physical mechanisms can, for example, create a stuck-at faults or timing faults in a circuit to change its behavior without changing its appearance. After employing these mechanisms to change the circuit behavior, the realized function of the circuit will differ from its apparent function. Because the modified circuit may violate common circuit

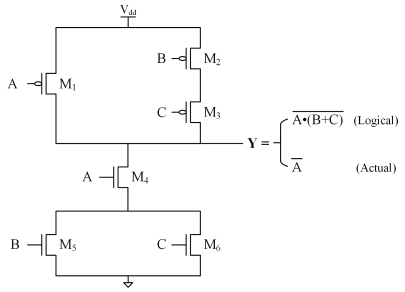

Fig. 8. An example of CMOS logic obfuscation.

truths, e.g., signal propagation and timing, a reverse engineer cannot successfully assume these truths to hold when trying to reconstruct the circuit function. If a reverse engineer can no longer exploit these simplifying assumptions, then his task will become significantly more challenging. In the following subsections, we present examples to show that different circuit implementation styles present different opportunities for obfuscation.

1) CMOS Circuits: CMOS circuits can be obfuscated to protect the design and confuse an attacker by using various device manipulations. For example, consider the circuit shown in Fig. 8 implementing the logic $Y = \overline{A \cdot (B + C)}$. Using the mechanisms described in Section V-A the circuit can be morphed to realize a logic different than its apparent one. If transistor M_5 and M_6 are converted from enhancement type to depletion type MOSFETs by doping, they will be always on. Similarly, if transistor M_2 is converted to an open circuit using source/drain doping, the node between M_2 and M_3 will be floating [37]. These changes would together cause the circuit to realize the functionality of an inverter $Y = \bar{A}$ instead of its apparent function of $Y = \overline{A \cdot (B + C)}$. Similarly, the circuit function could be modified by using the mechanisms described in Section V-B, such as via alteration, to change the behavior of the interconnects that drive the circuit inputs. For example, if *SA1* faults are created on the interconnects driving the B and C inputs, the circuit would again realize the function of an inverter, even if the transistors are unmodified.

2) Pass Transistors: Pass transistors are used as logic switches to create paths between the input and output nodes. Relative to CMOS logic, pass transistor logic is bidirectional and can be faster and reduce transistor count and power. Hence, pass transistor logic is often used in multiplexers, switches and efficient XOR implementations. The doping mechanisms from Section V-A allow for controlling the threshold voltage of pass transistors to create chosen stuck-at faults that modify circuit behavior.

Fig. 9 (a) shows cascaded logic using three pass transistors. It is possible to obfuscate the logic by using a high-threshold transistor for M_1 so that the voltage drop through the transistor is large enough for the cascade output to always be perceived as logic-0. This means that the input to the inverter at the end of the cascaded logic is always logic-0 and node Out realizes a stuck-at logic-1. The values applied to the gates of pass transistors M_2 and M_3 are irrelevant in this example, but this would not be apparent to the attacker.

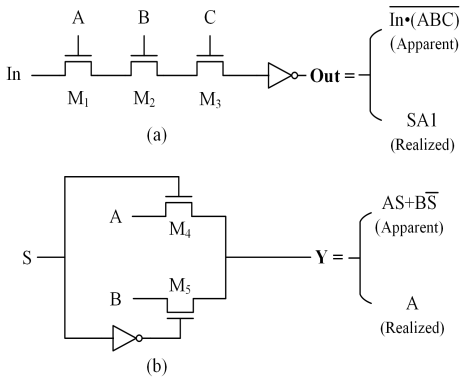

Fig. 9. Pass transistor circuits. (a) Cascaded pass transistors, (b) 2:1 Mux using pass transistors.

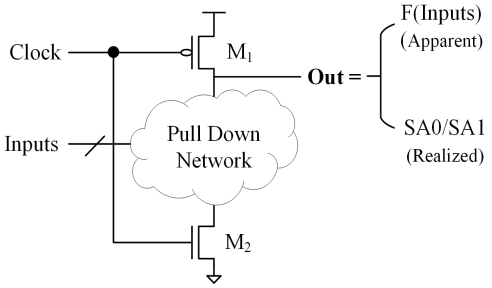

Fig. 10. Dynamic Logic.

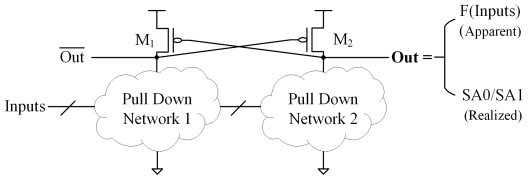

Fig. 11. Differential Cascode Voltage Switch (DCVS) Logic.

Fig. 9 (b) shows an apparent 2 : 1 multiplexer where one can similarly induce faults such that M_5 is always open and M_4 is always closed. The realized logic function after these changes would be $Y = A$ instead of a multiplexer.

3) Dynamic Logic and Differential Cascode Voltage Switch (DCVS) Logic: Dynamic logic and Differential Cascode Voltage Switch (DCVS) logic finds use in high-speed applications [59] where an NMOS pull-down network is preferable for performance reasons. Fig. 10 and Fig. 11 illustrate dynamic logic and DCVS logic, respectively. The two circuits as depicted are sensitive to changes in transistors M_1 and M_2, so a natural approach to obfuscation is to manipulate these transistors. For example these transistors can be made stronger or weaker using doping and channel length biasing as described in Section V-A, creating stuck-at faults at the outputs irrespective of the inputs. An attacker that cannot observe the properties of M_1 and M_2 would not be able to discern that the computational logic is irrelevant to the output value.

4) Flip-Flops: Large sequential digital systems are composed of combinational logic and state-holding elements such as flip-flops. A flip-flop circuit that may appear to be a

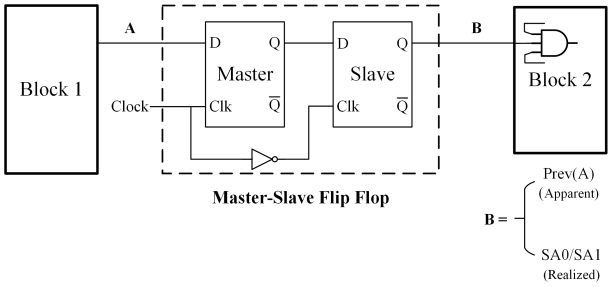

Fig. 12. Flip-flops under obfuscation.

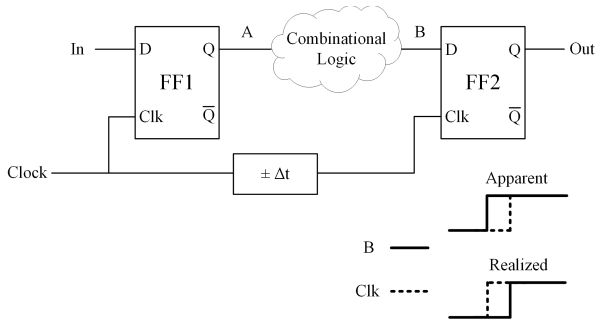

Fig. 13. Setup and Hold Violations illustration.

state-holding element can be manipulated to instead always hold a particular value. For example, consider the circuit shown in Fig. 12. Signal A is connected as the input to the flip-flop and from appearances would be expected to be sampled on the clock edge and passed on to the output of the flip-flop. In reality, by inducing a stealthy stuck-at fault, the output of the flip-flop can be assigned logic-0 or 1. For an attacker who tries to extract the logic, the circuit may appear to have a logical connection from Block 1 to Block 2 but the realized function has no such connection. Multiple such scenarios can be created with flip-flops that are inserted only for the purpose of increasing the complexity of reverse engineering the design.

a) Setup and hold violations: Synchronous sequential systems with flip-flops typically need to obey setup and hold time constraints to function as intended. Setup time requirements specify that flip-flop input signals must not change immediately before a clock edge, and hold time requirements specify that flip-flop input signals must not change immediately after a clock edge. Intentionally violating these requirements can create predictable errors that can be employed for obfuscation. For example, consider the circuit shown in Fig. 13. The path from FF_1 to FF_2 through the combinational logic has to obey the setup time requirements at FF_2 for a given clock period to operate without error, and a reverse engineer would typically assume this to be the case. If the devices and interconnects on a logic path were slowed down using techniques such as high-threshold transistors and interconnect thinning (Sec. V), the setup time would be violated, and the realized sequential behavior would differ from the apparent behavior. Similar to setup time violations, hold time violations can be induced but may require compound manipulations to speed up paths and induce errors.

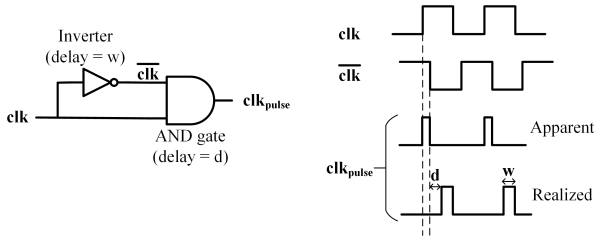

Fig. 14. Manipulating the clock signal used in Pulsed Latch.

b) Pulsed latch: Pulsed latch based flip-flop designs are common in high-performance systems where they are deployed to improve performance and reduce power dissipation [60]. Pulsed latches retain the advantage of both flip-flop and latch-based design. Instead of using master-slave latches with a clock, the pulsed latch has a single latch and uses a pulsed clock so that the latch is only open for a short time. The circuit to generate a pulsed clock, and the associated waveforms, are shown in Fig. 14. The behavior of the latch critically depends on the pulse width and timing, and one can subtly induce setup and hold time violations or race-through conditions by manipulating the pulsed clock. The width and timing of the pulsed clock can be manipulated by changing the delay of the inverter and the AND gate, respectively; slowing down the inverter increases the pulse width, and slowing down the AND gate will delay the arrival of the pulsed clock. An attacker would typically assume that a pulsed latch would be free of conditions such as flow-through, so these changes would cause the realized function of the latch to differ from the apparent one. Timing manipulations may be challenging for an attacker to detect during scan testing, because the scan chain is usually run at much slower speeds than the standard circuit operation.

B. Promoting Physical Mechanism to Logic Level

The previous subsection dealt with numerous ways to manipulate circuits, but now we focus specifically on gates that use a set of indistinguishable physical structures to implement different Boolean functions. An adversary observing this ambiguous physical structure will not know which logical variant of the gate he is seeing. Yet, it is important to note that, when using indistinguishable gate structures to realize multiple functions, the obfuscated gates will be larger in size and have a different pattern than the non-obfuscated gate variants that implement the same functions. So, if a design uses a mixture of obfuscated and non-obfuscated gates, then the attacker will likely have knowledge of which gates are obfuscated, which can help localize the uncertainty for reverse engineering. The reverse engineering problem becomes even more constrained when only a small number of gates are obfuscated [9] in this way. By contrast, for the mechanisms in the previous subsection, an attacker may have to consider every component in the design as potentially misleading.

1) Look-Up Tables: Any physical mechanism that creates obfuscated 0 or 1 constants can be used to program obfuscated look-up tables for fully programmable logic gates. An n-input look-up table requires 2^n programming bits to store the desired

output bit for every input combination. The programming bits would be implemented as nodes that are forced to be either stuck-at 1 or stuck-at 0 using one of the techniques from Sec. V. For example, the programming bit could have vias connecting to ground and vdd, and the attacker would have to determine which via is the dummy and which is real to know whether the corresponding gate output value was 0 or 1. An advantage of look-up tables is that they are fully general logic structures, but the disadvantage is that they are much larger than the CMOS logic gates that perform the same functions.

2) Programmable Logic Array (PLA) Cross Points: As with look-up tables, obfuscated connections can be used to program a PLA to resist reverse engineering. The PLA cross points that are connected would be indistinguishable from the open connections from the perspective of attacker.

3) Restricted Choice of Gate Functions: A less general alternative to look-up tables is to use gates that can implement just a few different logical functions, with the choice among the functions being determined by some hard to observe mechanism such as doped channel stops [30]. The work of Rajendran *et al.* [9] addresses the problem of how to use a small number of obfuscated gates to efficiently obfuscate an overall circuit function. Their work specifically considers a 2-input gate structure that has indistinguishable variants for implementing XOR, NAND, or NOR functions; learning which function the gate implements requires ability to distinguish between true vias and dummy vias. A similar style of obfuscated gate is proposed by Malik *et al.* [41].

C. Obfuscation Countermeasures: SAT Solving to Assist Reverse Engineering

Given a set of possible component implementations, and ability to query the obfuscated circuit to learn the correct computed values for any input, one can use an oracle-guided synthesis approach [61] to reconstruct the functionality of an obfuscated block of combinational logic. In practice, this procedure is quite effective due to the impressive capabilities of modern SAT solvers. SAT solving is used both for synthesizing new inputs that should be applied to the oracle, and is also used for deobfuscating the design once a sufficient set of input-output examples is obtained. For more details on this approach, we point interested readers to the work El Massad *et al.* [62]; their work shows that the logic function of a circuit with 200 obfuscated gates, each capable of implementing NAND/NOR/XOR (3^{200} possible logic functions), can be identified in less than an hour after observing just tens of input/output vectors [62]. Later work extends El Massad's attack using incremental SAT solvers for more efficient implementation [63]. This sort of attack can be applied to any context where the attacker knows a set of possible functions for each component, and can apply inputs and observe corresponding outputs through a scan chain. In certain types of obfuscated circuits, there is no attempt to hide the locations of components with obfuscated functions, and only the specific functions implemented by those components are secret. The ability to clearly identify obfuscated and non-obfuscated components helps to minimize the exponential increase in the space of possible circuit functions that must be implicitly searched during oracle-guided synthesis. We conjecture that low-level stealthy changes will be less susceptible to oracle-guided synthesis approaches because an attacker will not be able to identify *a priori* the non-obfuscated components in the circuit, and must therefore consider every component to be suspect yielding a much larger space of possible circuit functions that could be implemented.

VII. Discussion and Challenges

In above sections, various physical mechanisms that can be used for obfuscation were introduced. Below, we briefly discuss two related concepts, hardware Trojans and tamper protection techniques. We conclude this paper with a discussion of challenges and open research problems.

A. Hardware Trojans vs Obfuscation

While physical design obfuscation aims at *protecting* a design through various low level manipulations as discussed above, hardware Trojans can potentially exploit the same techniques to insert *malicious* functionality into a target circuit [37]. More concretely, what are sometimes called parametric Trojans [64] are quite similar to the mechanisms discussed in Secs. V and VI. There are, however, several important differences between Trojans and obfuscation. First, Trojans have malicious intent and are therefore inserted by adversaries in the design flow, while obfuscation is introduced by designers aiming to protect a circuit. Second, Trojans and obfuscation have vastly different reliability requirements. An obfuscated design is expected to work with high reliability, similar to any other circuit, or else the function implemented would be incorrect. In contrast, a low-reliability Trojan can still compromise the security of a system, even though its malicious intent, e.g., leaking of a cryptographic key, is not fully reliable. Third, often it must be possible to trigger a Trojan, e.g., through certain input values, a requirement which is entirely absent in the case of obfuscation. Finally, the requirement of being stealthy is often more crucial in the case of obfuscation, as the adversary specifically tries to overcome the stealthiness, something that is not always the case for hardware Trojans.

B. Tampering Protection

Hardware reverse engineering usually requires physically tampering with the target circuit. Obfuscation techniques, such as the ones described in this contribution, are one approach to provide on-chip protection by preventing the attacker from learning the logic function. Orthogonal to obfuscation techniques are tamper protections that prevent an attacker from even accessing data or silicon layers. Various countermeasures have been proposed to protect against physical access of circuits, and we mention here only a few of them. A prominent example is the IBM 4748 co-processor. It envelopes the hardware in a tamper-sensing and tamper-reactive shell along with tamper-triggers that can signal the processor to wipe all sensitive data and burn fuses [65]. While this can protect cryptographic keys, it does not prevent an attacker

from accessing the circuit to extract the IP. An approach to protecting the actual circuit is described in Mikulec *et al.* [66], based on a process that embeds gadolinium nitrate channels in silicon. If this layer were to be ground, it would explode and destroy nearby components. The assumption is that such a layer is placed at the top of the chip to provide protection against delayering.

A principle problem of physical tamper protection is that an attacker often has access to more than one chip. This allows him to devise strategies by running experiments which attempt to overcome the countermeasures. Even if several chips are destroyed, there is the risk that she eventually succceeds. The successful attack against a Trusted Platform Module (TPM) chip with strong tamper resistant features is an instructive example [67]. Nevertheless, for devices with low- to medium-security requirements, physical tamper protection can be an attractive approach. We note that tamper protection can be combined with the physical design obfuscation described in the contribution at hand.

C. Challenges

In this paper, we attempt to provide a systematic way to look at physical design obfuscation, together with several concrete examples for achieving low-level hardware obfuscation. As mentioned in the introduction, circuit obfuscation has only received scant attention in the scientific community and there are several challenging (and interesting) open research questions.

In the field of software, obfuscation techniques have been studied at depth and several complexity metrics have been developed. These metrics relate to *potency*, *stealth*, *performance* and resource *cost*. A good description of these can be found in Collberg *et al.* [68]. For hardware obfuscation, there are no metrics to quantify how well a circuit has been obfuscated. Often an adversary can "guess" or make conjecture about the functionality of a circuit. Such guesses may be based on design size, design density, power dissipation or any number of circuit characteristics that often characterize a specific function. Availability of diagnostic methods and commercial tools can lead an attacker to find the locations where a circuit may have been obfuscated. Availability of formal methods and tools may even allow an attacker to validate her hypothesis using measurements from the circuit. Most formal methods perform search in an exponential space. However, knowing that an adversary must search an exponential space to find the correct model says little about the practical hardness of his task. Current generation of Boolean SAT solvers demonstrate impressive capability in tackling large circuits. Thus, how well a circuit has been obfuscated is a complicated question that ensnares several different research questions into one. Quantifying the *hardness* of any obfuscation solution can go a long way towards engineering adoption.

Another important question is the *stealthiness* of the obfuscation, i.e., the device and logic-level techniques discussed in this paper. The strength of the methods heavily depends on the capabilities of the attackers: This is difficult to quantify yet important for determining the strength of an obfuscation techniques. There is a large design space for combinational and sequential circuits that lend themselves to the obfuscation methods discussed in this contribution. Little is known about optimum ways to design such circuits, which provide high security and at the same time keep the area and power overhead low. Related to this question is the CAD integration of obfuscation, i.e., an automated design flow which maps an arbitrary high-level design description to a circuit with strong reverse engineering resistance.

Another research direction is related to the practicality of any obfuscation solution. What is the impact on design cost (also known as non-recurrent engineering cost), schedule, manufacturing overhead, performance, area, and impact on yield? Practicality of the solutions also relates to ease of integration with existing tools and methodologies, or clear description of requirements to enable development of CAD tools for automatic obfuscation. It can be argued that this practical question is crucial for the use of physical design obfuscation in practice.

REFERENCES

[1] Rambus. *Use Cases: Personalization*, accessed on Feb. 17, 2016. [Online]. Available: http://www.rambus.com/use-cases-personalization/

[2] R. Torrance and D. James, "The state-of-the-art in semiconductor reverse engineering," in *Proc. 48th Design Autom. Conf.*, New York, NY, USA, 2011, pp. 333–338.

[3] K. Nohl *et al.*, "Reverse-engineering a cryptographic RFID tag," in *Proc. USENIX Secur. Symp.*, vol. 28. 2008, pp. 1–9.

[4] D. Strobel *et al.*, "Fuming acid and cryptanalysis: Handy tools for overcoming a digital locking and access control system," in *Proc. 33rd Ann. Cryptol. Conf.*, Aug. 2013, pp. 147–164.

[5] M. Kammerstetter, M. Muellner, D. Burian, C. Platzer, and W. Kastner, "Breaking integrated circuit device security through test mode silicon reverse engineering," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 549–557.

[6] S. Skorobogatov, "Physical attacks and tamper resistance," in *Introduction to Hardware Security and Trust*. New York, NY, USA: Springer, 2012, pp. 143–173.

[7] S. Skorobogatov and C. Woods, "Breakthrough silicon scanning discovers backdoor in military chip," in *Proc. 14th Int. Conf. Cryptograph. Hardw. Embedded Syst.*, 2012, pp. 23–40.

[8] M. Schobert, "Softwaregestütztes reverse-engineering Von Logikgattern in integrierten schhaltkreisen," Ph.D. dissertation, Dept. Math.-Naturwissenschaftliche Fakultät II, Inst. Inf. Humboldt Univ. Berlin, Berlin, Germany, Jun. 2011. [Online]. Available: http://www.degate.org/

[9] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security analysis of integrated circuit camouflaging," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 709–720.

[10] R. P. Cocchi, J. P. Baukus, L. W. Chow, and B. J. Wang, "Circuit camouflage integration for hardware IP protection," in *Proc. 51st ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2014, pp. 1–5.

[11] A. Akhunzada *et al.*, "Man-at-the-end attacks: Analysis, taxonomy, human aspects, motivation and future directions," *J. Netw. Comput. Appl.*, vol. 48, pp. 44–57, Feb. 2015.

[12] M. Rostami, F. Koushanfar, and R. Karri, "A primer on hardware security: Models, methods, and metrics," *Proc. IEEE*, vol. 102, no. 8, pp. 1283–1295, Aug. 2014.

[13] Chipworks. (2012). *Intel Xeon E3-1230V2 CPU Ivy Bridge Tri-Gate 22 nm Layout and DFM Feature Analysis*, accessed on Aug. 4, 2015. [Online]. Available: https://www.chipworks.com/TOC/Intel_Ivy_Bridge_22nm_CPU_Layout_CWR-1204%-801_TOC.pdf

[14] T. Sugawara *et al.*, "Reversing stealthy dopant-level circuits," in *Cryptographic Hardware and Embedded Systems–CHES*. Berlin, Germany: Springer, 2014, pp. 112–126.

[15] J. C. Tsang, J. A. Kash, and D. P. Vallett, "Picosecond imaging circuit analysis," *IBM J. Res. Develop.*, vol. 44, no. 4, pp. 583–603, Jul. 2000.

[16] K. Nikawa, "Laser-SQUID microscopy: Novel nondestructive and non-electrical-contact tool for inspection, monitoring and analysis of LSI-chip-electrical-defects," in *Proc. Int. Microprocess. Nanotechnol. Conf.*, Oct./Nov. 2001, pp. 62–63.

[17] J. D. Parham and Y. Kim, "Hiding circuit components using boundary blurring techniques," in *Proc. IEEE Annu. Symp. VLSI*, Dec. 2010, pp. 5–7.

[18] J. T. McDonald, Y. Kim, and D. Koranek, "Deterministic circuit variation for anti-tamper applications," in *Proc. 7th Annu. Workshop Cyber Secur. Inf. Intell. Res.*, 2011, Art. no. 68.

[19] J. T. McDonald, J. Todd, Y. C. Kim, and M. R. Grimaila, "Protecting reprogrammable hardware with polymorphic circuit variation," in *Proc. 2nd Cyberspace Res. Workshop*, 2009, pp. 63–78.

[20] J. T. McDonald and A. Yasinsac, "Program intent protection using circuit encryption," in *Proc. 8th Int. Symp. Syst. Inf. Secur.*, 2006, pp. 1–9.

[21] J. B. Wendt and M. Potkonjak, "Hardware obfuscation using PUF-based logic," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2014, pp. 270–277.

[22] Y. M. Alkabani and F. Koushanfar, "Active hardware metering for intellectual property protection and security," in *Proc. 16th USENIX Secur. Symp.*, 2007, pp. 20:1–20:16.

[23] Y. Alkabani, F. Koushanfar, and M. Potkonjak, "Remote activation of ICs for piracy prevention and digital right management," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2007, pp. 674–677.

[24] L. Li and H. Zhou, "Structural transformation for best-possible obfuscation of sequential circuits," in *Proc. IEEE Int. Symp. Hardw.-Oriented Secur. Trust (HOST)*, Jun. 2013, pp. 55–60.

[25] R. S. Chakraborty and S. Bhunia, "HARPOON: An obfuscation-based SoC design methodology for hardware protection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 10, pp. 1493–1502, Oct. 2009.

[26] J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: Ending piracy of integrated circuits," in *Proc. Design, Autom. Test Eur. (DATE)*, Mar. 2008, pp. 1069–1074.

[27] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Logic encryption: A fault analysis perspective," in *Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, Mar. 2012, pp. 953–958.

[28] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *Proc. 49th ACM/EDAC/IEEE Design Autom. Conf.*, Jun. 2012, pp. 83–89.

[29] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *Proc. IEEE Int. Symp. Hardw.-Oriented Secur. Trust (HOST)*, May 2015, pp. 137–143.

[30] J. Baukus, L. W. Chow, and W. M. Clark, "Digital circuit with transistor geometry and channel stops providing camouflage against reverse engineering," U.S. Patent 5 783 846, Jul. 21, 1998.

[31] J. P. Baukus, L. W. Chow, and W. M. Clark, "Digital circuit with transistor geometry and channel stops providing camouflage against reverse engineering," U.S. Patent 5 930 663, Jul. 27, 1999. [Online]. Available: http://www.google.com/patents/US5930663

[32] R. P. Cocchi, J. P. Baukus, B. J. Wang, L. W. Chow, and P. Ouyang, "Building block for a secure CMOS logic cell library," U.S. Patent 8 111 089, Feb. 7, 2012. [Online]. Available: http://www.google.com/patents/US8111089

[33] L. W. Chow, J. P. Baukus, B. J. Wang, and R. P. Cocchi, "Camouflaging a standard cell based integrated circuit," U.S. Patent 8 151 235, Apr. 3, 2012. [Online]. Available: http://www.google.com/patents/US8151235

[34] M. Tehranipoor and C. Wang, *Introduction to Hardware Security and Trust*. New York, NY, USA: Springer, 2011.

[35] B. Colombier and L. Bossuet, "Survey of hardware protection of design data for integrated circuits and intellectual properties," *IET Comput. Digit. Techn.*, vol. 8, no. 6, pp. 274–287, Nov. 2014.

[36] U. Guin, K. Huang, D. DiMase, J. M. Carulli, M. Tehranipoor, and Y. Makris, "Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain," *Proc. IEEE*, vol. 102, no. 8, pp. 1207–1228, Aug. 2014.

[37] G. T. Becker, F. Regazzoni, C. Paar, and W. P. Burleson, "Stealthy dopant-level hardware trojans," in *Cryptographic Hardware and Embedded Systems–CHES*. Berlin, Germany: Springer, 2013, pp. 197–214.

[38] J. Kao, A. Chandrakasan, and D. Antoniadis, "Transistor sizing issues and tool for multi-threshold CMOS technology," in *Proc. 34th Annu. Design Autom. Conf.*, Jun. 1997, pp. 409–414.

[39] SypherMedia. (2012). *SypherMedia Library–Circuit Camouflage Technology*, accessed on Apr. 21, 2015. [Online]. Available: http://www.smi.tv/SMI_SypherMedia_Library_Intro.pdf

[40] J. P. Baukus, W. M. Clark, Jr., L.-W. Chow, and A. R. Kramer, "Secure integrated circuit," U.S. Patent 6 294 816, Sep. 25, 2001. [Online]. Available: http://www.google.com/patents/US6294816

[41] S. Malik, G. T. Becker, C. Paar, and W. P. Burleson, "Development of a layout-level hardware obfuscation tool," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2015, pp. 204–209.

[42] X. Chen, S. Li, and J. Zhang, "A contrast between rule-based and model-based dummy metal fill in ASIC design," in *Proc. IEEE Int. Conf. Intell. Control Inf. Process. (ICICIP)*, Aug. 2010, pp. 601–606.

[43] D. O. Ouma, "Modeling of chemical mechanical polishing for dielectric planarization," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, 1998.

[44] V. B. Suresh, P. V. Kumar, and S. Kundu, "On lithography aware metal-fill insertion," in *Proc. IEEE 13th Int. Symp. Quality Electron. Design (ISQED)*, Mar. 2012, pp. 200–207.

[45] J. P. Baukus, L.-W. Chow, W. M. Clark, Jr., and P. O. Yang, "Multilayered integrated circuit with extraneous conductive traces," U.S. Patent 6 924 552, Aug. 2, 2005. [Online]. Available: http://www.google.com/patents/US6924552

[46] V. S. Shilimkar, S. G. Gaskill, and A. Weisshaar, "Closed-form expressions for modeling metal fill effects in interconnects," in *Proc. 15th IEEE Workshop Signal Propag. Interconnects (SPI)*, May 2011, pp. 97–100.

[47] Y. Kim, D. Petranovic, and D. Sylvester, "Simple and accurate models for capacitance increment due to metal fill insertion," in *Proc. Asia South Pacific Design Autom. Conf.*, 2007, pp. 456–461.

[48] S. Raghavan, I. Schmadlak, and S. K. Sitaraman, "Interlayer dielectric cracking in back end of line (BEOL) stack," in *Proc. IEEE 62nd Electron. Compon. Technol. Conf. (ECTC)*, May/Jun. 2012, pp. 1467–1474.

[49] L.-W. Chow, J. Baukus, and W. Clark, "Integrated circuits protected against reverse engineering and method for fabricating the same using an apparent metal contact line terminating on field oxide," U.S. Patent 2002 0 096 776, Jul. 25, 2002. [Online]. Available: http://www.google.com/patents/US20020096776

[50] L.-W. Chow, J. P. Baukus, and W. M. Clark, Jr., "Integrated circuits protected against reverse engineering and method for fabricating the same using vias without metal terminations," U.S. Patent 6 791 191, Sep. 14, 2004. [Online]. Available: http://www.google.com/patents/US6791191

[51] H. Kim *et al.*, "Novel flowable CVD process technology for sub-20nm interlayer dielectrics," in *Proc. IEEE Int. Interconnect Technol. Conf. (IITC)*, Jun. 2012, pp. 1–3.

[52] K. Kinoshita *et al.*, "Via-shape-control for copper dual-damascene interconnects with low-k organic film," *IEEE Trans. Semicond. Manuf.*, vol. 21, no. 2, pp. 256–262, May 2008.

[53] H.-Y. Chen and Y.-W. Chang, "Routing for manufacturability and reliability," *IEEE Circuits Syst. Mag.*, vol. 9, no. 3, pp. 20–31, Aug. 2009.

[54] S. Pujari, R. M. Smey, T. Yan, H. H. Madden, and P. H. Madden, "Interconnect synthesis for lithography and manufacturability in deep submicron design," in *Proc. 12th Workshop Synth. Syst. Integr. Mixed Inf. Technol. (SASIMI)*, Oct. 2004, pp. 1–8.

[55] L. S. Melvin, III, J. P. Mayhew, B. D. Painter, and L. D. Barnes, "Assist feature placement analysis using focus sensitivity models," in *Proc. 22nd Eur. Mask Lithograph. Conf. (EMLC)*, 2006, pp. 1–7.

[56] N. Dhumane, S. K. Srivathsa, and S. Kundu, "Lithography constrained placement and post-placement layout optimization for manufacturability," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2011, pp. 200–205.

[57] P. Gupta, A. B. Kahng, and C.-H. Park, "Detailed placement for enhanced control of resist and etch CDs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 12, pp. 2144–2157, Dec. 2007.

[58] X. Shi, S. Hsu, J. F. Chen, C. M. Hsu, R. J. Socha, and M. V. Dusa, "Understanding the forbidden pitch phenomenon and assist feature placement," *Proc. SPIE*, vol. 4689, pp. 985–996, Jul. 2002.

[59] D. Somasekhar and K. Roy, "Differential current switch logic: A low power DCVS logic family," *IEEE J. Solid-State Circuits*, vol. 31, no. 7, pp. 981–991, Jul. 1996.

[60] Y. Shin and S. Paik, "Pulsed-latch circuits: A new dimension in ASIC design," *IEEE Design Test Comput.*, vol. 28, no. 6, pp. 50–57, Nov./Dec. 2011.

[61] S. Jha, S. Gulwani, S. A. Seshia, and A. Tiwari, "Oracle-guided component-based program synthesis," in *Proc. ACM/IEEE 32nd Int. Conf. Softw. Eng.*, vol. 1. May 2010, pp. 215–224.

[62] M. E. Massad, S. Garg, and M. V. Tripunitara, "Integrated circuit (IC) decamouflaging: Reverse engineering camouflaged ICs within minutes," in *Proc. 22nd Annu. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, San Diego, CA, USA, Feb. 2015, pp. 1–14.

[63] D. Liu, C. Yu, X. Zhang, and D. Holcomb, "Oracle-guided incremental SAT solving to reverse engineer camouflaged logic circuits," in *Proc. Design Autom. Test Eur. (DATE)*, Mar. 2016, pp. 433–438.
[64] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design Test Comput.*, vol. 27, no. 1, pp. 10–25, Jan./Feb. 2010.
[65] J. G. Dyer, M. Lindemann, R. Perez, R. Sailer, L. van Doorn, and S. W. Smith, "Building the IBM 4758 secure coprocessor," *Computer*, vol. 34, no. 10, pp. 57–66, Oct. 2001.
[66] F. V. Mikulec, J. D. Kirtland, and M. J. Sailor, "Explosive nanocrystalline porous silicon and its use in atomic emission spectroscopy," *Adv. Mater.*, vol. 14, no. 1, pp. 38–41, 2002.
[67] C. Tarnovsky, "Deconstructing a 'secure' processor," in *Proc. Black Hat DC*, vol. 2010. 2010, pp. 1–6. [Online]. Available: http://blackhat.com/presentations/bh-dc-10/Tarnovsky_Chris/BlackHat-DC-2010-Tarnovsky-DASP-slides.pdf
[68] C. Collberg, C. Thomborson, and D. Low, "A taxonomy of obfuscating transformations," Dept. Comput. Sci., Univ. Auckland, Auckland, New Zealand, Tech. Rep. 148, Jul. 1997.

Arunkumar Vijayakumar (S'16) is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Massachusetts Amherst, Amherst, MA, USA, under the supervision of Prof. S. Kundu. His research interests include VLSI CAD, VLSI circuit design, and hardware security.

Vinay C. Patil (S'16) is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Massachusetts Amherst, Amherst, MA, USA, under the supervision of Prof. S. Kundu. His research interests include VLSI circuit design, hardware security, machine learning, and neuromorphic computing.

Daniel E. Holcomb (M'07) received the B.S. and M.S. degrees in electrical and computer engineering from the University of Massachusetts Amherst, Amherst, MA, USA, and the Ph.D. degree in electrical engineering and computer sciences from the University of California at Berkeley, Berkeley, CA, USA. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, University of Massachusetts Amherst. His research interests include span formal verification, very large-scale integration, embedded systems, and hardware security.

Christof Paar (F'11) received the M.Sc. degree from the University of Siegen, and the Ph.D. degree from the Institute for Experimental Mathematics, University of Essen, Germany. He was the Chair of Embedded Security with Ruhr-Universität Bochum, Bochum, Germany, and is an Affiliated Professor with the University of Massachusetts Amherst, Amherst, MA, USA. He cofounded with C. Koc, the Workshop on Cryptographic Hardware and Embedded Systems series. He has authored over 150 peer-reviewed publications and is coauthor of the textbook *Understanding Cryptography* (New York, NY, USA: Springer-Verlag, 2010). He is a cofounder of ESCRYPT–Embedded Security, a leading consultancy firm in applied security that is currently part of Bosch. His research interests include implementation techniques for cryptography, hardware security, physical security, and security evaluation of real-world systems.

Sandip Kundu (F'07) is a Professor with the University of Massachusetts Amherst. Prior to joining academia, he spent several years in industry as a Research Staff Member with IBM Research Division and then with Intel Corporation as a Principal Engineer. He has authored or coauthored over 200 research papers in VLSI design and test, and holds several key patents, including ultradrowsy sleep mode in processors, and has given over a dozen tutorials at various conferences. He is a Fellow of the Japan Society for Promotion of Science, Senior International Scientist of the Chinese Academy of Sciences, and a Distinguished Visitor of the IEEE Computer Society. He is currently an Associate Editor of the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING. He was served as an Associate Editor of the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON VLSI SYSTEMS, and the *ACM Transactions on Design Automation of Electronic Systems*. He has been Technical Program Chair/General Chair of multiple conferences, including ICCD, ATS, ISVLSI, DFTS, and VLSI Design Conference.