Georg T. Becker, Marc Fyrbiak, Christian Kison

Hardware Obfuscation

April 12, 2017

Springer

Chapter 1
Hardware Obfuscation: Techniques and Open Challenges

1.1 Introduction

There are many applications for IC reverse-engineering. While there are legitimate reasons for IC reverse-engineering, some have malicious intend such as as IP infringement and technological espionage. Particularly, Intellectual Property (IP) theft and counterfeit products are a major challenge for the industry. In many cases, the initial step in counterfeiting or stealing of IP is to reverse-engineer a chip or IP core in order to integrate the IP into one's own design illegitimately. Hence, there are various reasons why hardware companies demands obfuscation methods to hamper reverse-engineering of their designs. For security-critical devices reverse-engineering can also be a potential attack vector. An adversary can leverage reverse-engineering to disclose internal details of the design in order to enable further attacks on the system. For example, implementation attacks such as side-channel or fault attacks exploit implementation structures and thus an attacker gaining knoledge of the used implementation and countermeasures gains a significant attack advantage. In addition to these malicious goals, reverse-engineering can also be used to detect patent infringements and IP theft as well as to identify Hardware Trojans.

As a consequence, hardware obfuscation techniques that hamper reverse-engineering are of great interest. In this chapter, we present and discuss state-of-the-art hardware obfuscation techniques at two distinct levels. Hardware obfuscation at the layout level targets the extraction of the device's netlist. To be more precise, the underlying principle is to prevent the distinct identification of combinatorial gates. In Section 1.2, we provide a summary of the proposed layout-level obfuscation techniques and additionally a security evaluation. However, not every case of IP piracy starts with reverse-engineering of the targeted Integrated Circuit (IC). For example, most IP provides do not manufacture their own chips, but only sell IP cores in the form of hard and soft IP cores. In these scenarios, the adversary is already in possession of the

netlist (without the need of IC reverse-engineering).
In order to prevent the disclosure of internal details, obfuscation transformations at the netlist level are required. In Section 1.3, we present and discuss the state-of-the-art proposed netlist-level obfuscation methods and automatic reverse-engineering capabilities. Furthermore, we address various limitations and open challenges for the different netlist-level obfuscation techniques.

1.2 Layout-Level Obfuscation

The first step in reverse-engineering of a targeted Application Specific Integrated Circuit (ASIC) is to obtain precise images of each chip's layer and subsequently to identify the individual gates and their connectivity. Based on this information the netlist of the design can be derived which enables the reverse-engineer to analyze the design as well as to make copies of it. Hence, the principal goal of layout-level obfuscation is to hamper this netlist disclosure by use of special combinatorial gates that cannot be correctly identified via visual reverse-engineering techniques by using a Scanning Electron Microscope (SEM). Layout-level obfuscation has been of interest in the industry for many years – the first patents date back to the 1980s [Pec86]. However, despite the industry's ongoing interest in this topic, it has been largely ignored by the the scientific community and only recently the first works have appeared [RSSK13, CBCW14, SHF14, MBPB15].

In the following, we introduce several proposals of how layout level obfuscation can be realized. Particularly, we illustrate the so-called *camouflage gates* or *look-alike gates* that are utilized instead of standard cells in order to prevent the visual identification of the implemented logic function. These camouflage gates are the main building block in layout level hardware obfuscation

1.2.1 Camouflage gates

The main idea of camouflage gates is to hide the logic function of the gates in design layers that are hard to detect visually. The main assumption of most camouflage gates is that metal layers and polysilicon layers are easily recognizable using visual reverse-engineering techniques such as SEMs. Hence, the goal of building camouflage gates is to create gates that are identical at these layers but still have a different functionality. The most common way to achieve this is to only change the dopant masks, i.e., to build gates that are identical on all design layers and only differ in the dopant polarity in some active areas. For example, this technique is used in [CBCW14, SHF14, MBPB15]. A different approach is to use a mixture of real and dummy contacts to camouflage the functionality of the obfuscated gates [RSSK13]. The main idea behind this is that extra effort is needed to reverse-engineer the contacts. However, compared to dopant-based obfuscation, contact-based obfuscation is considerably easier to reverse-engineer. A more detailed analysis of the difficulty of reverse-engineering both dopant-based and contact-based obfuscation – as well as no obfuscation at all – is provided in Section 1.2.4.

Obfusgate camouflage gates: In the following the main idea behind dopant-based camouflage gates is explained using the example of a camouflage gate called *Obfusgate*, which has been proposed in [MBPB15]. The heart of the Obfusgate is a so-called *Obfuscell*. Depending on the dopant polarity in its active areas, an Obfuscell can be configured to be either an inverter, a buffer (input=output), or to output a constant '1' or '0'. Figure 1.2(a) shows the layout of an Obfuscell. It has three active areas A1, A2, and A3. The dopant polarity within these areas define the configuration. To create a buffer for example, the input needs to be connected to the output via A2 as illustrated in Figure 1.2(b). For example, the active area A2 can be used to build a direct connection between the input and output. This is achieved by doping the entire active area A2 positively. On the other hand, if the middle region of A2 is doped negatively, this creates a p-n-p junction and hence a diode in cut-off. This is depicted in Figure 1.3(a). Hence, by using different dopant polarity in the active area, one can connect or disconnect inputs. All dopant-based camouflage gates are based on this main idea [CBCW14, SHF14, MBPB15]. In the Obfuscell design the active areas A1 and A3 have the layout of a pmos and nmos transistor respectively (see Figure 1.3(b)) for the doping of a pmos transistor) and hence can also create connections to the output. Figure 1.2(b) shows the four different possible configurations of the Obfuscell. An inverter is configured by using A1 and A3 as pmos and nmos transistors respectively and disabling the connection in A2 as described above. If the Obfuscell is connected as a buffer, area A2 is doped positively and hence a connection is formed. Simultaneously, the two transistors in A1 and A3 needs to be "disabled", i.e., their outputs should be floating. How this can be done is depicted in Figure 1.3(d) at the example a pmos transistor (area A1). The source contact which is connected to VDD is doped negatively instead of positively which basically creates a well-contact and a n-p junction to the output, i.e., a diode in cut-off. Hence, the output (drain) is floating and therefore the transistor is "disabled". Similarly, Figure 1.3(c) shows how the drain can be connected to the input (connected to VDD) by doping the entire region positively. This way the active area A1 can be used to set the output to a constant one as needed for the "always 1" configuration (see Figure 1.2(b) for details).

Hence, depending on the dopant polarity in A1, A2 and A3, the Obfuscell is either an inverter, a buffer (i.e., input = output), a constant '1' or a constant '0'. This Obfuscell is then used as a building block to build Obfusgates that form the obfuscated standard cell library. Figure 1.1 depicts the structure of an "Obfusgate". It consists of five "Obfuscells" and a 4-input NAND gate.

Depending on the configuration of the Obfuscells (i.e. on their dopant), the Obfusgate can implement many different logic functions. For example, configuring all Obfuscells as buffers results in a four input AND gate. Configuring the four Obfuscells that are used as inputs as inverters on the other hand results in a four input NOR gate. In total, 162 different configurations, each with a unique logic behavior, can be realized with an Obfusgate as depicted in Figure 1.1. Furthermore, by setting an Obfuscell that is connected to an input of the Obfusgate to a constant ’1’, this input has effectively been turned into a “dummy input”. Setting two input Obfuscells to a constant ’1’ and the other Obfuscells to a buffers results in an two input AND gate. The two inputs that are set to a constant ’1’ are the “dummy inputs” since the signal connected to these inputs has no effect on the output. Therefore, any signal can be connected to such an input, creating “dummy wires”. Note that an attacker cannot distinguish between a dummy input and a regular input and hence this technique can increase the obfuscation significantly.

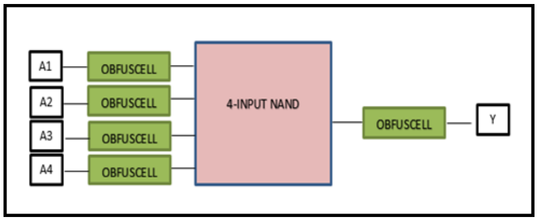
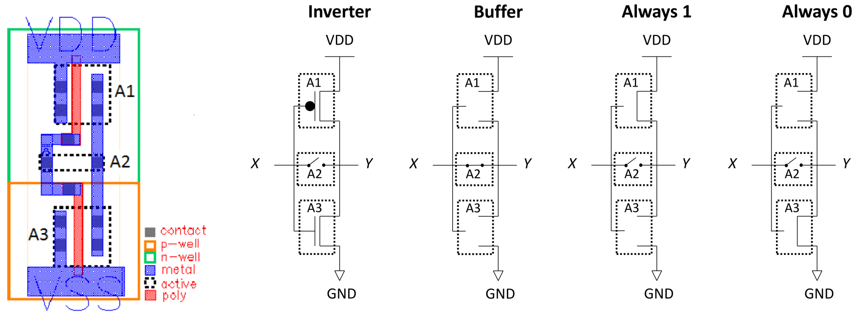


Fig. 1.1 Schematic of a single Obfusgate that consists of 5 Obfuscells together with a 4-input NAND gate. Depending on the configuration of the Obfuscells, the Obfusgate can realize 162 different logic functions.

DPD-LUT camouflage gates: Shiozaki, Hori and Fujino [SHF14] used a different approach to build dopant-based camouflage gates. Their design is based on a 2-bit look-up table (LUT) similar to the LUTs used in FPGA designs. The input to these LUTs are special read-only memory cells called *Diffusion Programmable ROM* (DP-ROM) that are “programmed” using the dopant polarity. They function in exactly the same way as in the active area A2 of the Obfuscell and depicted in Figure 1.3(a). Each DP-ROM cell consists of two of these active areas and depending on the configuration of the dopant the output of the cell is either connected to VDD or GND.

The camouflage gate, which is called Diffusion Programmable Device Lookup Table (DPD-LUT), is depicted in Figure 1.4. A DPD-LUT can be configured to any 2-input logical function, i.e., it can realize $2^4 = 16$ different functions. We would also like to note that this design approach is not restricted to 2-input LUTs. In an analogous manner, it is possible to build a 3-input DPD-LUT or, as in FPGAs, 4- or 5-input LUTs. Using larger LUTs will likely result in a larger overhead but also in a higher grade of obfuscation. The



(a) (b)

Fig. 1.2 a) Layout view of the Obfuscell which has three active regions A1, A2 and A3 whose dopant polarity defines the logic function of the gate. The gate can be configured as a inverter, buffer, “always 1” or “always 0” gate as shown on the right side (b).

smallest overhead could probably be achieved by combining DPD-LUTs of different sizes. However, the distribution of differently sized DPD-LUTs might help an attacker gain some insight into the obfuscated design. Questions like this have not been researched yet, and hence it is not clear to what extent the combination of different look-alike gates into one common obfuscated standard cell library can decrease obfuscation strength.

SMI’s approach: Cocchi *et al.* proposed two different strategies to build camouflage gates [CBCW14]. The first one is to construct custom camouflage gates whose logic function is hard to reverse-engineer but which are easily identifiable as camouflage gates, similar to the approaches of Malik *et al.* and Shiozaki *et al.* Unfortunately, no details are provided on how this is achieved and how many different functions one look-alike gate can implement. The second strategy proposed is to use existing standard cells and to only modify a few in order to create a new functionality. Since these modifications are hard to detect, a reverse-engineer will mistake the camouflage gate for a "‘normal"’ standard cell and hence will come up with a faulty netlist. The advantage of this approach is that only a few camouflage gates might offer enough obfuscation in certain situations, while greatly reducing the introduced overhead. However, the disadvantage is that a single camouflage gate offers less obfuscation since it usually can only realize very few different logic functions. How exactly Cocchi *et al.* modified the standard cells and how many different functions such a cell can implement has not not been disclosed. However, this approach shares many similarities with the dopant-level hardware Trojans presented at CHES 2013 [BRPB13], which also change the functionality of standard cells while making the detection of these modifications as hard as possible. Basically, the gates are modified as also done in the Obfuscell by connecting outputs to VDD or ground and removing transistors as depicted

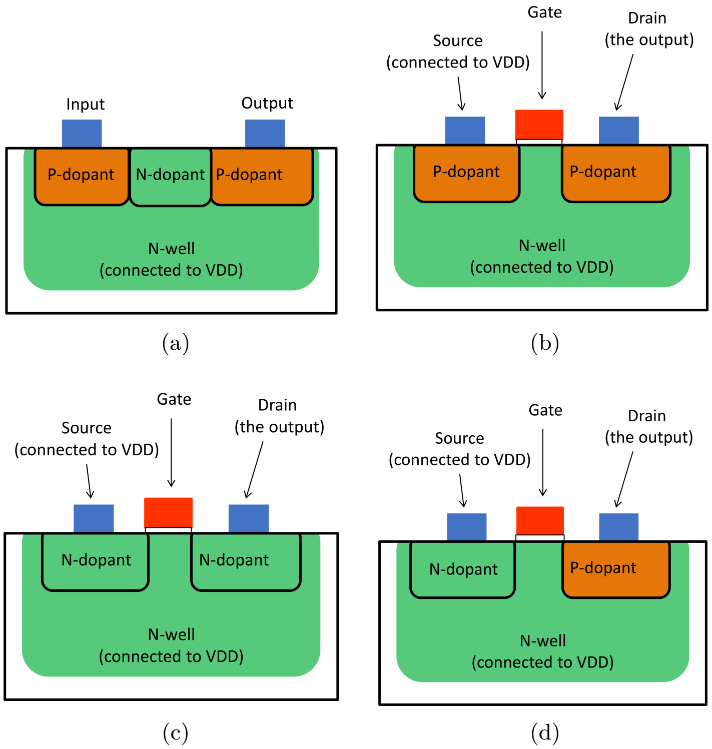


Fig. 1.3 Cross section view of the active areas. In (a) active area A2 is depicted configured in “cut-off’, i.e., the input is disconnected from the output. By also doping the middle part positively the configuration is changed to a direct connection between the input and output. Active area A1 can be used as a pmos transistor as depicted in b). In c) the active area A1 is doped positively instead of negatively which results in a constant connection between input (connected to VDD) and the output. How a floating output can be realized for A1 is shown in d). Only the source region of A1 is doped negatively which results in a n-p junction between source and drain, i.e., a diode in cut-off.

in Figure1.3. Hence, there is a large overlap in the construction of camouflage gates and stealthy layout-level hardware Trojans, and the Obfusgate design was inspired by these Trojans.

1.2.2 Obfuscating the Connectivity:

Camouflage gates obfuscate the logic function of individual gates. However, they do not conceal the connectivity, i.e., a reverse-engineer can still see which gates are connected with each other. Ideally, a reverse-engineer should not infer the function of a block after it has been obfuscated. But the con-

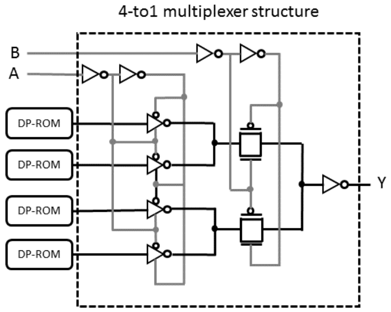
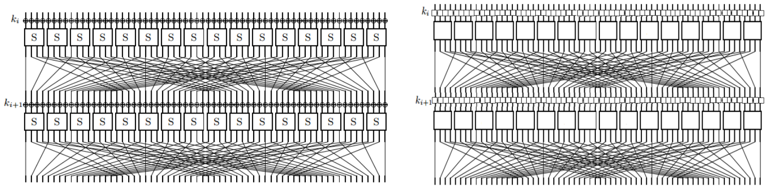


Fig. 1.4 The structure of a DPD-LUT. It consists of four DP-ROM cells and a 2-input multiplexer structure with the inputs A and B and the output Y. Any 2-bit logical function can be implemented with such a DPD-LUT cell.

nectivity of the individual gates reveals a lot of useful information to a reverse-engineer. This was illustrated in [MBPB15] with the example of the block cipher PRESENT [BKL$^+$07]. The PRESENT round function is depicted in Figure 1.5(a). The key insight is that when camouflage gates are used, the logic of the gates are not known but. When grouping cells that are connected with each other together, the resulting graph would look like Figure 1.5(b). The white boxes represent blocks whose logic function is not known to the reverse-engineer due to the use of camouflage gates. However, since the connections are known, it is not difficult to e.g. to identify the 4-bit SBoxes used in PRESENT, since they are functions with four inputs and exactly four outputs. The fact that 4-input functions as e.g. 8-input functions reveals a lot of information to a reverse-engineer about the employed encryption function.

Hence, since a lot of information is not obfuscated, camouflage gates by themselves are not enough to prevent reverse-engineering if the attacker's goal is to collect information about the design structure or to identify the location of certain IP blocks within a chip. In order to solve this problem, the Obfusgate library heavily uses "dummy wires" that conceal the connectivity. More than half of the obfuscation gates in the AES SBox and PRESENT round functions originally are 2-input gates. Each Obfusgate that is configured as a 2-input gate has two dummy inputs and hence also two dummy wires. In the proof-of-concept implementation of the Substitution and Permutation Layer of PRESENT, 941 dummy wires and 1103 normal wires are used[MBPB15] This very large amount of dummy wires effectively hides the connectivity and hence the structure of the design since an attacker cannot differentiate between dummy wires and real wires. However, it is important to note that this level of obfuscation comes with a large area overhead. Furthermore, the current version randomly connects the dummy wires. While this works for small designs such as the PRESENT round function, for larger designs the routing overhead would increase to a level that would make routing impossible. Hence, just random connections do not scale for large designs. Thus, how to efficiently obfuscate connectivity information at the layout level is an interesting open research problem. In general, the addition of dummy wires or connections can also be achieved using by the camouflage gates proposed in [CBCW14, SHF14, RSSK13]. This can be realized by inserting additional gates that only have the purpose of creating dummy wires. Again, the optimum number of additional gates and how to integrate them has not been analyzed and is therefore an open question.



(a) PRESENT round function

(b) PRESENT round function with obfuscated combinatorial gates

Fig. 1.5 a) Figure of the PRESENT round function, taken from [BKL$^+$07]. b) When the combinatorial gates are replaced with camouflage gates, a reverse-engineer does not know the logic function of the SBoxes any longer. The structure of the round function on the other hand is still clearly visible due to the wires connecting the individual blocks and registers.

1.2.3 Further obfuscation techniques

Besides using camouflage gates, other techniques to hamper reverse-engineering at the layout level have been proposed. For example, reverse-engineering non-volatile memory can be more difficult as reverse-engineering combinatorial memory [QCF$^+$16]. The idea is therefore to not implement the entire design using normal combinatorial gates but also include non-volatile memory cells that are programmed after manufacturing. The content of these memory cells then determine the logic behavior of the chip. For example, this technique could be combined with DPD-LUT camouflage gates: Instead of using DP-ROM cells that are programmed based on dopant polarity, other non-volatile memory cells that are programmed after manufacturing can be used. One advantage of using non-volatile memory is that this also prevents the factory from over-producing the ICs. After manufacturing the fabricated chips are non-functional and hence only the IP owner can program and hence activate the chips.

Another technique which makes layout level reverse-engineering is the use of special filler cells [CBCW14]. Typically, in a digital chip there often gaps between gates due to routing constrains etc. These gabs are usually filled with so called “filler cells” to fulfill certain design rules. These filler cells can be easily identified as non-functional cells during reverse-engineering. The idea is to instead use cells that look like legitimate gates, i.e., replace the non-functional filler cells with (non-functional) camouflage gates. Since a reverse-engineer does cannot easily distinguish these cells from functional camouflage gates, this can significantly increase the required reverse-engineering effort.

1.2.4 Reverse-engineering camouflage gates

Ideally, camouflage gates make it impossible to reverse-engineer the gates using visual techniques by optical means. Having a process to determine a precise doping level and impurity is of outmost importance for the chip production and their failure analysis. Special processes are necessary to measure impurities and dopants. Therefore, the dopant-based and via-based camouflage gates do not prevent reverse-engineering in general, but rather hamper the process. In this section, we briefly discuss several reverse-engineering techniques that are able to reveal the functionality of the proposed camouflage gates. Notably these techniques emerged from failure analysis, trying to locate faults in dopant concentrations, defects, or impurities.

Delayering and Hardware Reverse Engineering: The art of Hardware Reverse engineering begins at the Printed Circuit Board (PCB) and package level of the IC piece of hardware. First the IC is cropped out or de-soldered from the PCB. Please note that this step is non-trivial for some flip-chip packages with underfill. The challenge to protect the die is becoming ever more difficult with reduced die size and thickness. Secondly the package has to be removed by wet-chemical or mechanical means. Hereby, again, the die is to be protected from any harm which often results in choosing wet-chemical depackaging, as the die is protected by the seal-layer[1] from the front side. The backside offers enough silicon in the bulk to withstand carefully applied depackaging processes as well. The bonding wires are of special concern, as newer copper bondings are, compared to gold bonding wires, hard to preserve. For the invasive hardware reverse engineering the wires can be neglected once their connectivity is known or the connectivity can be derived. Advanced techniques for finding bonding wire connectivity can be done by (3D) X-Ray or selective packaging delayering with a mill.

Once the die is fully recovered, the die is alternately delayered and digitalized by optical means or in a SEM/Focused Ion Beam (FIB). The following delayering processes are, again, a combination of different wet-chemical and mechanical polishing. Here it is of outmost importance for the quality of the process, to handle the equipment in an experienced way. Planarization of the

[1] passivation, often SiO_2

current layer with a huge surface to thickness ratio is one, if not the hardest challenge to master. Please note knowing your Region of Interest (ROI) comes very handy at this point, as the planar surface can be reduced significantly. The reverse engineer can pinpoint his ROI while neglecting the rest of the chip [KFP15]. Different metals and glasses have to be investigated and selectively removed without destroying functional information of the IC [QCF+16].

Digitalizing and imaging is done in a SEM or FIB derivate in current state-of-the art reverse engineering. With modern technologies sizes hitting the diffraction limit of optical microscopes, are more advanced visualizing tools mandatory. One the one hand this has the drawback of a moderate investment, but on the other hand can result in smaller images when the color information from optical images drop out .During the image acquiring a brightness yield from the metals, to the vias and a brightness difference to the background is created due to different substance (electrical-)properties. A clear brightness yield from the SEM/FIB images is beneficial for the post-processing as it allows to distinguish between vias, wires and Spin-on dielectric (SOD), shown in figure 1.6.

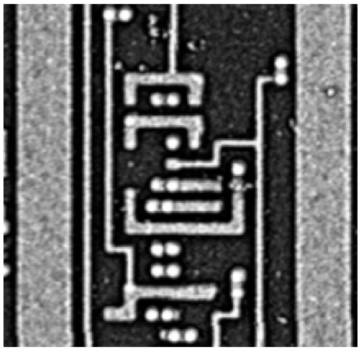


Fig. 1.6 The brightness allows to distinguish between wires, vias and the SOD. Metal 1 in an older technology is shown. The brighter dots are vias between Metal1 and Metal2.

Post-processing is done in software after every layer has been digitalized in tile images. The tile images are stitched, vectorized and finally reverse engineered to get their functional interpretation. This is a very tedious and repetitive task that can be (semi-)automated to support the reverse engineer. Different approaches for post-processing are out of scope of this work.

Voltage Contrast: By exploiting the very nature of n-wells and p-wells, a reverse-engineer can observe a brightness yield from secondary electrons or ions by using a SEM or FIB [PDH+11]. Particularly, Sugawara *et al.* [SSF+14] demonstrate the use of Voltage Contrast (VC) to distinguish the vias connectivity with a clear brightness yield 1.7. Notably, it is not trivial in practice to obtain meaningful results from the brightness yield, especially if the ROI is large (in the worst-case the ROI covers the whole chip). In the event the layer images raise doubts, a reverse-engineer can enhance the doping contrast [RP05]. The VC shown in by Sugawara *et al.* [SSF+14] can be automatically included during the delayering process with a SEM, which is a state-of-the-art hardware reverse-engineering equipment due to the shrinking technology size.

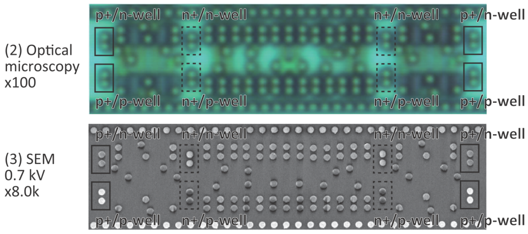


Fig. 1.7 Reversing stealthy dopant-level circuits. A brightness yield indicates the possible dopant regions. Taken from [SSF+14]

Chemical etching and staining Distinguishing the dopant characteristics is often accompanied by measurement of the dopant concentration. This technique is commonly employed in failure analysis and quality control processes of silicon wafer vendors. Based on chemical etch rates or chemical staining, the dopant area can be distinguished [RP05, pro, Bec98]. For example, the chemical optimal *dash etching* exhibits different colors of p-regions and n-regions, cf. Figure 1.8. An overview of different chemical recipes and practical approaches is given by Beck [Bec98].

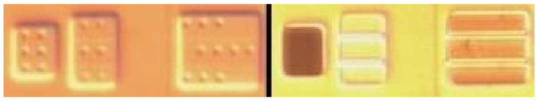


Fig. 1.8 Dash Etching. The right picture shows a CMOS cell with p^+ dopant regions, stained with a blue/green effect depending on the applied etching time and the dopant concentration. Figure taken from [pro]

It is noteworthy that a major drawback of chemical dash etching is the optical equipment. This limits the reverse-engineer to large areas, due to the optical diffraction limit. Particularly, might become a challenge for future shrinking technology sizes, where the (stealthy) dopant areas shrink with the cell size. Advanced techniques to measure etchant rates, e.g with a SEM should be considered.

Scanning Microscopy: In order to detect single point defects or local faults covering a few atoms of impurities, mainly two techniques for lateral doping sensitivity profiling have been established: Scanning Capacitance Microscopy (SCM) and Scanning Spreading Resistance Microscopy (SRRM). While SCM is based on capacitance differences in the substrate, SRRM is derived from the Atomic Force Microscopy (AFM) [Sch06]. As a consequence of their small-area approaches and the required equipment, they are not recommended for identification of stealthy dopant areas. They are capable to do so, but take alot of time. Nevertheless they are listed for the sake of completeness.

1.3 Netlist-Level Obfuscation

The successful extraction of a chip's netlist has various implications ranging from counterfeiting/cloning to technology espionage, cf. Section 1.1. However in several scenarios, the adversary possesses the netlist in form of hard or soft IP cores or an untrusted foundry obtains the netlist via the chip's blueprint. To counteract IP piracy, several counterfeit avoidance methods such as secure split test and the use of Physical Unclonable Functions (PUFs) were proposed. Additionally, watermarking and IP protection schemes are a related strand of research, however this work focuses on netlist reverse-engineering and netlist obfuscation.

Adversary Model: Before presenting the details regarding netlist-level reverse-engineering and obfuscation transformations, we briefly recap the adversary model in this scenario. We assume that the adversary has access to the flattened gate-level netlist without any a priori high-level information such as synthesis options or hierarchy structures. The high-level adversarial goal can be coarsely defined as information disclosure of how a design works in detail, in order to leverage further attacks.

1.3.1 Netlist Reverse-Engineering Techniques

In the following we summarize the state-of-the-art in the field of algorithmic reverse-engineering of gate-level netlists. A discussion of the available techniques is vital in order to analyze the strength of an obfuscation technique. Furthermore, an overview of all published methods supports the classification of a reverse-engineering task by means of time.

In 1999, Hansen *et al.* [HYH99] reported several strategies for a human reverse-engineer to extract high-level information from the ISCAS-85 benchmark suite. The strategies include the identification of common library components such as decoders or adder units and the analysis of repeated modules such as in data-path circuits. Shi *et al.* [STGR10] introduced a technique to algorithmically extract Finite State Machines (FSMs) from a flattened netlist based on their inherent implementation structure. In particular, FSMs are detected based on an *enable tree* as well as *strongly connected component* identification approach. This technique is employed in the subsequent work of Shi *et al.* [SGR+12], where the netlist (with eliminated FSM) is analyzed and its functional modules extracted. In 2012, a technique for matching an unknown subcircuit against abstract library components was introduced by Li *et al.* [LWS12]. The technique is based on pattern mining of the simulation traces as well as model checking. The subsequent work by Li *et al.* in 2013 [LGS+13] identified word-level structures which provides a more abstract, high-level view of the design. This work is furthermore taken as a basis for algorithmic component identification such as counters, register files, and adders etc. [STP+13, STL+14]. A challenging task for functional identification is the potentially permuted input mapping of the reference circuit and the design under investigation. Gascón *et al.* [GSD+14] addressed this problem with a template-based approach in 2014.

After this brief recap of (semi-)algorithmic reverse-engineering capabilities, we highlight various obfuscation techniques. Furthermore, we discuss their advantages and limitations according to the published reverse-engineering techniques. First, we present control and data flow obfuscation strategies and secondly reconfiguration-based methods.

1.3.2 Control Flow Obfuscation

A notable challenge from a reverse-engineer's point of view is to make sense of the design's control flow to disclose information how different modules interact with each other. Control flow obfuscation refers to a set of transformations to hamper this analysis, particularly by modification of an FSM.

Hardware Metering refers to a conglomeration of tools and security protocols to enable the design house the post manufacturing control of a produced device. In particular, this methodology introduced in 2001 allows an unique way to identify each IC by a passive or active fingerprint [KQ01]. Note that this strand of research is also related to obfuscation as the identification circuitry should be hard to reverse-engineer.

Internal active hardware metering can be seen as a form of control flow obfuscation [AK07]. The original FSM of the design is augmented by several states, cf. Fig. 1.9. In particular, the initial value of the FSM registers is determined by the output of a PUF. Only if the correct input sequence (and thus the correct traversal of the FSM states) is given, the augmented FSM ends up in the initial state of the original FSM and hence the design operates correctly. A key feature of this technique is that the number of Flip Flops (FFs) influences the number of possible states exponentially and hence provides a lightweight solution to the issue of an unique IC identifier.

A related technique is combinatorial locking (external active hardware metering). This technique extends combinatorial logic networks with the addition of XOR / XNOR nodes [RKM08] or the gate is hidden in reconfigurable logic [BTZ10]. Only if the correct key value is applied to the input of the added nodes, the circuit is equivalent to the original one. However the claimed security of several subsequent works in this field is challenged by the recent work of Subramanyan *et al.* [SRM15]. The proposed attack is based on satisfiability checking that practically unlocked the vast majority of allegedly locked designs. A detailed summary and discussion of the diverse hardware metering techniques is outside of the scope of this work and hence the interested reader is referred to [Kou12].

A similar FSM-based obfuscation technique to provide anti-piracy features such as authentication was proposed by Chakraborty *et al.* in 2008 [CB08]. An FSM is added to the circuitry whose inputs are the primary design inputs and it has one output. Furthermore this output is XORed with a few selected nodes of the design. Consequently, the FSM outputs a logical one as long as the correct input sequence is not applied to the primary input and only for the correct sequence the FSM transits into the state that outputs a logical zero, so that the design is equivalent to the unobfuscated one. In a subsequent work [CB09a], the FSM output is extended by an additional signal that represents the output of a logical OR of the input variables. Later on the method was applied to Register Transfer Level (RTL) via synthesis, application of the obfuscation on netlist-level, and subsequent decompilation to generate the obfuscated RTL [CB09b].

Limitations: All denoted techniques have the fundamental limitation

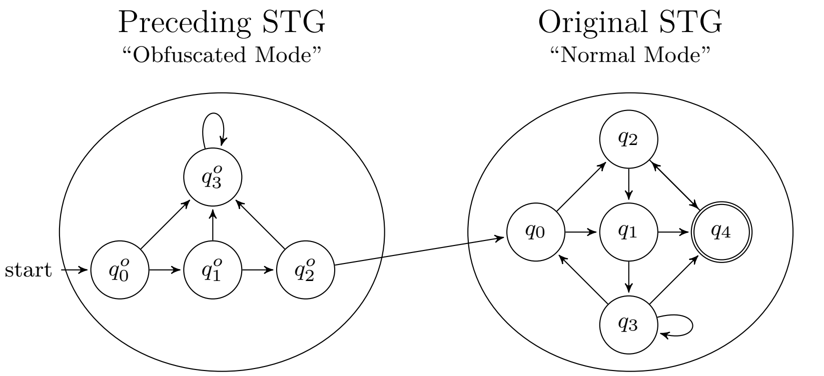


Fig. 1.9 Example: FSM Obfuscation with preceding State Transition Graph (STG) based on Fig. 1a in [CB09b]

that merely the control flow (via the FSM) is obfuscated, but the inherent circuit structure for subcomponents is preserved (even if gates are appended to the output of combinatorial subcomponents). Thus, a practical evaluation is vital regarding the influence of the different automatic techniques in presence of control flow obfuscation. Furthermore, all enumerated techniques should be evaluated regarding the FSM reverse-engineering technique by Shi *et al.* [STGR10]. Particularly, all security analyses do not address the issue of reverse-engineering from the last state of the obfuscation circuitry that transits to the original initial state of the design to the best of the authors knowledge. As the design is somehow locked for an invalid input sequence, an adversary would search for conditions such as multiplexers or enable signals where a meaningful output is generated (or at least a larger set of node influences the primary output), e.g., based on established techniques such as SAT solvers. Depending on the state transition graph, an inversion may result in an exponential number of input pattern candidates, however the complexity should be evaluated practically.

Another fundamental limitation is the structure of the obfuscation circuitry itself. For example, Chakraborty *et al.* [CB09a] utilize a special enable signal in their technique. First, the signal that enables the correct behavior has a large fan-out cone and its target gates are XOR elements. Second, each node in the set of selected nodes where an XOR gate is added to the output is chosen by a metric. Such inherent structures leak information regarding the implementation and might be identified using methods such as pattern matching or SAT solvers. Overall, all enumerated techniques should be evaluated regarding the statements in the limitation as well as the FSM reverse-engineering technique by Shi *et al.* [STGR10].

1.3.3 Combined Data and Control Flow Obfuscation

To address the fundamental limitations of control flow obfuscation transformations, several works combined the FSM-based alteration with data flow obfuscation to generated malformed output instead of locking the device as described in the following.

In 2010 Chakraborty *et al.* [CB10] presented a technique that partially consists of the prior outlined FSM alteration. Particularly, the authors demonstrated how the FSM can be interwoven with the design in order to hamper isolation of the FSM. In addition to the FSM obfuscation, the data flow is obfuscated through generation of phony output, if the system is not in a valid state (depending on the primary input sequence). This is realized by assignment of different arithmetic/logical functions to the output of the obfuscated module.

A further combined obfuscation transformation was presented by Li *et al.* in 2013 [LZ13]. The key element of their methodology is the incorporation of the entire design and not only the FSM. Thus, also general circuitry such as adders or memory circuitry is transformed by the obfuscation. To be more precise, the obfuscation strategy is based on several methods that moves registers in sequential circuitry, encodes the circuit with a bijective function that is applied before and after the register stage, and addition of logic conditions under that a register value is updated. Sergeichik *et al.* adapted the concept of opaque predicates for hardware in 2014 [SI14]. The underlying principle of opaque predicates is to generate a constant output during runtime in order to hamper static analyses. Particularly, the authors insert special constant generating circuitry on the Hardware Description Language (HDL)-level, e.g., an Linear Feedback Shift Register (LFSR) where all FF values are zero or a latch-based circuit.

Limitations: Although the combination of control and data flow obfuscation definitely increase the reverse-engineer's efforts, the denoted obfuscation circuitries are static by nature. If the reverse-engineer makes sense of a structural obfuscated subcircuit, then this subcircuit will not change its functionality at some subsequent point in time. Notably, these described obfuscation techniques focus on ASICs and not on field-programmable hardware such as Field Programmable Gate Arrays (FPGAs). Similar to the control flow obfuscation transformations, the proposed techniques were not evaluated regarding publicly known algorithmic reverse-engineering approaches, cf. Section 1.3.1. The decrease of information disclosure by these automatic techniques could improve the justification for the proposed obfuscation transformations.

1.3.4 Reconfiguration Obfuscation

In order to change the designs appearance during runtime, several works exploit reconfiguration features to obfuscate a design. Notably, this methodology requires runtime field-programmable hardware features, however it addresses the generic limitation of the prior described techniques.

Porter *et al.* proposed an obfuscation transformation based on dynamic polymorphic reconfiguration in 2009 [PSK+09]. The underlying principle is the gate replacement implemented by the dynamic reconfiguration feature of FPGAs as well as Look-up tables (LUTs). In particular, the different gates are realized by different configurations of the LUTs. Furthermore, signals are added to the design in order to hide function signatures. To preserve the semantic of the obfuscated function, a recovery key is utilized and subsequently added to the output of the reconfigured circuit. In 2013 Gören *et al.* extended an FSM-based obfuscation technique (see Section 1.3.2) with PUFs and a dynamic reconfiguration scheme, in order to provide a low-cost FPGA bitstream protection [GOY+13]. The FSM state transition depends several PUF instances that are implemented in distinct partial configuration bitstreams reconfigured during runtime. Depending on the stored PUF outputs, the design is either locked or unlocked.

Limitations: The reconfiguration features provide a significant advantage as the adversary has to reverse-engineer has to analyze a reconfigurable part for each partial design. However, the work by Porter *et al.* can be simulated and thus reverse-engineered (certainly with increased efforts). Similarly to the control-flow based obfuscation, the work by Gören *et al.* suffers from the generic limitation that only the FSM is transformed by the obfuscation (and the rest of the design remains unchanged).

1.4 Conclusion

Hardware obfuscation techniques are demanded by the industry to hamper IP piracy and technological espionage. Particularly, obfuscation transformation aim to increase the adversary's efforts in reverse-engineering a target device or design. In this chapter, we addressed hardware obfuscation at the layout levels as well as the netlist level.In terms of layout level obfuscation relatively few public information is available. While obfuscation is been in use for many years, how exactly — and how effectively — it is being used is not discussed publicly. Only very recently were the first scientific paper published in that regard. Most of these layout level obfuscation techniques are based on the idea to construct camouflage gates based on changes of the dopant polarity in the active area. However, several visual reverse-engineering techniques exists that can detect the dopant polarity in an active area. These techniques require additional steps and equipment compared to traditional layout reverse-engineering and hence can make reverse-engineering considerably harder. However, none of the layout level obfuscation techniques can completely prevent reverse-engineering. In general, form a research perspective, many unanswered questions remain in this area. Furthermore, the fact that the companies that specialize on reverse-engineering do not reveal their techniques to the public, estimating the cost of reverse-engineering a design with and without layout obfuscation is currently very difficult.

Several works proposed diverse methods to realize obfuscation transformations on the netlist level ranging from control flow-based techniques to reconfiguration-based methods. However, we identified various limitations for the different approaches especially regarding the security considerations. The coarse adversary model for obfuscation should be regarded in detail with respect to the system model and the defensive goal. Particularly, reverse-engineering of a design in order to disclose IP and patching of a design in order to eliminate locking features are elementary different goals. Furthermore, a fundamental issue for the majority of the analyzed works is the omission of the automatic reverse-engineering techniques, cf. Section 1.3.1. Particularly, an evaluation of the diverse obfuscation transformations combined with the reverse-engineering techniques is viable for future research in this area. Additionally to the obfuscation transformations, the reverse-engineering techniques have to be further explored in order to improve both the obfuscation transformations as well as the modeling of real-world adversarial capabilities.

Overall, hardware obfuscation provides a powerful set of tools to increase an adversary's reverse-engineering efforts. To really understand the level of obfuscation and security achieved by the different techniques it is also crucial to understand the capabilities of reverse-engineers. Unfortunately, often the public knowledge of the state-of-the-art reverse-engineering techniques is limited since reverse-engineering companies do not publish their methods. In many cases the real advantage of the different obfuscation technologies are therefore hard to estimate in practice. In general, there are still more open then solved research questions in the area of hardware obfuscation.

References

[AK07] Yousra Alkabani and Farinaz Koushanfar. Active hardware metering for intellectual property protection and security. In *Proceedings of the 16th USENIX Security Symposium, Boston, MA, USA, August 6-10, 2007*, 2007.

[Bec98] Friedrich Beck. *Integrated Circuit Failure Analysis: A Guide to Preparation Techniques*. Wiley-Interscience, 1998.

[BKL$^+$07] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe. Present: An ultra-lightweight block cipher. In *Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems (CHES 07)*, pages 450–466, Berlin, Heidelberg, 2007. Springer-Verlag.

[BRPB13] G. T. Becker, F. Regazzoni, C. Paar, and W. P. Burleson. Stealthy dopant-level hardware trojans. In *Cryptographic Hardware and Embedded Systems (CHES 2013)*, LNCS. Springer, 2013.

[BTZ10] Alex Baumgarten, Akhilesh Tyagi, and Joseph Zambreno. Preventing IC piracy using reconfigurable logic barriers. *IEEE Design & Test of Computers*, 27(1):66–75, 2010.

[CB08] Rajat Subhra Chakraborty and Swarup Bhunia. Hardware protection and authentication through netlist level obfuscation. In *2008 International Conference on Computer-Aided Design, ICCAD 2008, San Jose, CA, USA, November 10-13, 2008*, pages 674–677, 2008.

[CB09a] Rajat Subhra Chakraborty and Swarup Bhunia. HARPOON: an obfuscation-based soc design methodology for hardware protection. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 28(10):1493–1502, 2009.

[CB09b] Rajat Subhra Chakraborty and Swarup Bhunia. Security through obscurity: An approach for protecting register transfer level hardware IP. In *IEEE International Workshop on Hardware-Oriented Security and Trust, HOST 2009, San Francisco, CA, USA, July 27, 2009. Proceedings*, pages 96–99, 2009.

[CB10] Rajat Subhra Chakraborty and Swarup Bhunia. RTL hardware IP protection using key-based control and data flow obfuscation. In *VLSI Design 2010: 23rd International Conference on VLSI Design, 9th International Conference* *on Embedded Systems, Bangalore, India, 3-7 January 2010*, pages 405–410, 2010.

[CBCW14] Ronald P. Cocchi, James P. Baukus, Lap Wai Chow, and Bryan J. Wang. Circuit camouflage integration for hardware ip protection. In *Proceedings of the 51st Annual Design Automation Conference (DAC 14)*, pages 153:1–153:5, New York, NY, USA, 2014. ACM.

[GOY$^+$13] Sezer Gören, Ozgur Ozkurt, Abdullah Yildiz, H. Fatih Ugurdag, Rajat Subhra Chakraborty, and Debdeep Mukhopadhyay. Partial bitstream protection for low-cost fpgas with physical unclonable function, obfuscation, and dynamic partial self reconfiguration. *Computers & Electrical Engineering*, 39(2):386–397, 2013.

[GSD$^+$14] Adria Gascón, Pramod Subramanyan, Bruno Dutertre, Ashish Tiwari, Dejan Jovanovic, and Sharad Malik. Template-based circuit understanding. In *Formal Methods in Computer-Aided Design, FMCAD 2014, Lausanne, Switzerland, October 21-24, 2014*, pages 83–90, 2014.

[HYH99] Mark C. Hansen, Hakan Yalcin, and John P. Hayes. Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering. *IEEE Design & Test of Computers*, 16(3):72–80, 1999.

[KFP15] Christian Kison, Jürgen Frinken, and Christof Paar. *Cryptographic Hardware and Embedded Systems – CHES 2015: 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, chapter Finding the AES Bits in the Haystack: Reverse Engineering and SCA Using Voltage Contrast, pages 641–660. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[Kou12] Farinaz Koushanfar. *Introduction to Hardware Security and Trust*, chapter Hardware Metering: A Survey, pages 103–122. Springer New York, New York, NY, 2012.

[KQ01] Farinaz Koushanfar and Gang Qu. Hardware metering. In *Proceedings of the 38th Design Automation Conference, DAC 2001, Las Vegas, NV, USA, June 18-22, 2001*, pages 490–493, 2001.

[LGS$^+$13] Wenchao Li, Adria Gascón, Pramod Subramanyan, Wei Yang Tan, Ashish Tiwari, Sharad Malik, Natarajan Shankar, and Sanjit A. Seshia. Wordrev: Finding word-level structures in a sea of bit-level gates. In *2013 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2013, Austin, TX, USA, June 2-3, 2013*, pages 67–74, 2013.

[LWS12] Wenchao Li, Zach Wasson, and Sanjit A. Seshia. Reverse engineering circuits using behavioral pattern mining. In *2012 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2012, San Francisco, CA, USA, June 3-4, 2012*, pages 83–88, 2012.

[LZ13] Li Li and Hai Zhou. Structural transformation for best-possible obfuscation of sequential circuits. In *2013 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2013, Austin, TX, USA, June 2-3, 2013*, pages 55–60, 2013.

[MBPB15] S. Malik, G.T. Becker, C. Paar, and W.P. Burleson. Development of a layout-level hardware obfuscation tool. In *VLSI (ISVLSI), 2015 IEEE Computer Society Annual Symposium on*, pages 204–209, July 2015.

[PDH$^+$11] Seth Prejean, Brennan Davis, Lowell Herlinger, Richard Johnson, Renee Parente, and Mike Santana. Special techniques for backside deprocessing. In *Microelectronics Failure Analysis: Desk Reference*. A S M International, 2011.

[Pec86] Henry Pechar. Circuit to prevent pirating of an mos circuit, April 15 1986. US Patent 4,583,011.

[pro] Silicon Pr0n: silicon just the way you like it. Last visited Febr. 2016, http://siliconpr0n.org/wiki/doku.php?id=start.

[PSK$^+$09] Roy Porter, Samuel J. Stone, Yong C. Kim, J. Todd McDonald, and LaVern A. Starman. Dynamic polymorphic reconfiguration for anti-tamper circuits. In

19th International Conference on Field Programmable Logic and Applications, FPL 2009, August 31 - September 2, 2009, Prague, Czech Republic, pages 493–497, 2009.

[QCF+16] S. E. Quadir, J. Chen, D. Forte, N. Asadizanjani, S. Shahbazmohamadi, L. Wang, J. Chandy, and M. Tehranipoor. A survey on chip to system reverse engineering. *ACM Journal on Emerging Technologies in Computing Systems (JETC), to apear*, PP(??):??, 2016.

[RKM08] Jarrod A. Roy, Farinaz Koushanfar, and Igor L. Markov. EPIC: ending piracy of integrated circuits. In *Design, Automation and Test in Europe, DATE 2008, Munich, Germany, March 10-14, 2008*, pages 1069–1074, 2008.

[RP05] Erwan Le Roy Robert Pajak, Frank Baiocchi. dopant imaging on front surface of silicon devices with a coaxial photon-ion column. In *ISTFA 2005: Proceedings of the 31st International Symposium for Testing and Failure Analysis*, ISTFA. PROCEEDINGS. A S M International, 2005.

[RSSK13] Jeyavijayan Rajendran, Michael Sam, Ozgur Sinanoglu, and Ramesh Karri. Security analysis of integrated circuit camouflaging. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 709–720. ACM, 2013.

[Sch06] Dieter K. Schroder. *Semiconductor Material and Device Characterization*. Wiley-Interscience, 2006.

[SGR+12] Y. Shi, B. H. Gwee, Ye Ren, Thet Khaing Phone, and Chan Wai Ting. Extracting functional modules from flattened gate-level netlist. In *Communications and Information Technologies (ISCIT), 2012 International Symposium on*, pages 538–543, 2012.

[SHF14] Mitsuru Shiozaki, Ryohei Hori, and Takeshi Fujino. Diffusion programmable device : The device to prevent reverse engineering. *IACR Cryptology ePrint Archive*, 2014:109, 2014.

[SI14] Vladimir Sergeichik and Alexander Ivaniuk. Implementation of opaque predicates for fpga designs hardware obfuscation. *Journal of Information, Control and Management Systems*, 12(2), 01/2014.

[SRM15] Pramod Subramanyan, Sayak Ray, and Sharad Malik. Evaluating the security of logic encryption algorithms. In *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2015, Washington, DC, USA, 5-7 May, 2015*, pages 137–143, 2015.

[SSF+14] Takeshi Sugawara, Daisuke Suzuki, Ryoichi Fujii, Shigeaki Tawa, Ryohei Hori, Mitsuru Shiozaki, and Takeshi Fujino. Reversing stealthy dopant-level circuits. In *Cryptographic Hardware and Embedded Systems (CHES 2014)*, volume 8731 of *LNCS*, pages 112–126. Springer, 2014.

[STGR10] Yiqiong Shi, Chan Wai Ting, Bah-Hwee Gwee, and Ye Ren. A highly efficient method for extracting fsms from flattened gate-level netlist. In *International Symposium on Circuits and Systems (ISCAS 2010), May 30 - June 2, 2010, Paris, France*, pages 2610–2613, 2010.

[STL+14] Pramod Subramanyan, Nestan Tsiskaridze, Wenchao Li, Adria Gascón, Wei Yang Tan, Ashish Tiwari, Natarajan Shankar, Sanjit A. Seshia, and Sharad Malik. Reverse engineering digital circuits using structural and functional analyses. *IEEE Trans. Emerging Topics Comput.*, 2(1):63–80, 2014.

[STP+13] Pramod Subramanyan, Nestan Tsiskaridze, Kanika Pasricha, Dillon Reisman, Adriana Susnea, and Sharad Malik. Reverse engineering digital circuits using functional analysis. In *Design, Automation and Test in Europe, DATE 13, Grenoble, France, March 18-22, 2013*, pages 1277–1280, 2013.