Noname manuscript No.
(will be inserted by the editor)

A Survey of Algorithmic Methods in IC Reverse Engineering

Leonid Azriel · Julian Speith · Nils Albartus · Ran Ginosar · Avi Mendelson · Christof Paar

Accepted: 13 July 2021

Abstract The discipline of reverse engineering integrated circuits (ICs) is as old as the technology itself. It grew out of the need to analyze competitor's products and detect possible IP infringements. In recent years, the growing hardware Trojan threat motivated a fresh research interest in the topic. The process of IC reverse engineering comprises two steps: *netlist extraction* and *specification discovery*. While the process of netlist extraction is rather well understood and established techniques exist throughout the industry, specification discovery still presents researchers with a plurality of open questions. It therefore remains of particular interest to the scientific community. In this paper, we present a survey of the state of the art in IC reverse engineering while focusing on the specification discovery phase. Furthermore, we list noteworthy existing works on methods and algorithms in the area and discuss open challenges as well as unanswered questions. Thereby, we observe that the state of research on algorithmic methods for specification discovery suffers from the lack of a uniform evaluation approach. We point out the urgent need to develop common research infrastructure, benchmarks, and evaluation metrics.

Leonid Azriel
Technion - Israel Institute of Technology, Haifa, Israel
E-mail: leonida@tx.technion.ac.il

Julian Speith
Max Planck Institute for Security and Privacy, Bochum, Germany
Ruhr University Bochum, Bochum, Germany
E-mail: julian.speith@rub.de

Nils Albartus
Max Planck Institute for Security and Privacy, Bochum, Germany
Ruhr University Bochum, Bochum, Germany
E-mail: nils.albartus@rub.de

Ran Ginosar
Technion - Israel Institute of Technology, Haifa, Israel
E-mail: ran@ee.technion.ac.il

Avi Mendelson
Technion - Israel Institute of Technology, Haifa, Israel
E-mail: avi.mendelson@tce.technion.ac.il

Christof Paar
Max Planck Institute for Security and Privacy, Bochum, Germany
Ruhr University Bochum, Bochum, Germany
E-mail: christof.paar@csp.mpg.de

1 Introduction

Reverse engineering of integrated circuits serves a multitude of purposes [32]. For example, understanding the details of a competitor's IC helps to conduct a competitive analysis. Furthermore, patent infringements can be detected by locating the stolen IP in a competitor's IC. In addition, a particularly growing topic in recent years is the detection of hardware Trojans — a process that requires comprehensive retrieval of functionality. Additional applications of IC reverse engineering include the detection of counterfeit devices, failure analysis, and monitoring of semiconductor suppliers.

The very concept of reverse engineering may be perceived as an indignant peeking. Therefore, it is natural to question the ethics of reverse engineering in general as well as its compliance with the law. In the US, the legitimacy of reverse engineering was established by the US Semiconductor Chip Protection Act of 1984, to which most industrialized countries subscribe by now. Specifically, it allows reverse engineering of commercial semiconductor products for educational purposes. Another issue in that regard is the usage of the obtained information, which is clearly limited by law [30].

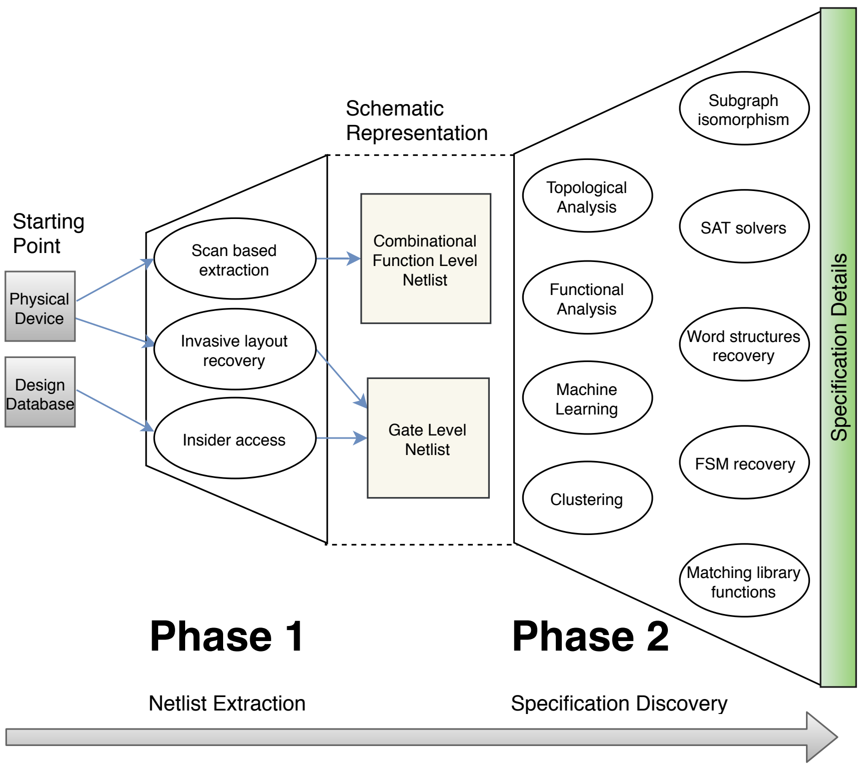

Fig. 1 Flow diagram of the hardware reverse engineering process comprising two major phases: (1) netlist extraction and (2) specification discovery. The second phase includes a variety of methods and algorithms that are not necessarily mutually exclusive.

On the technical side, reverse engineering of Integrated Circuits (ICs) is a complex process that involves multiple disciplines and skills. The process usually takes a physical device as input and outputs a human-readable specification. Note that in this work the term *specification* is used in a broader context, generally referring to the objective of the reverse engineering process. Along the way, this process comprises various steps, which can be divided into two distinct phases (*cf.* Figure 1). During the first phase, *netlist extraction*, the IC is analyzed to generate a gate-level netlist description. Traditionally, netlist extraction involves a series of invasive techniques, such as package removal and delayering [76,31,50]. Though this method prevails, there are alternative ways to extract the netlist. For example, scan chains used for production testing in virtually any modern digital IC provide an easy access to the circuit internals and enable learning the functions of the logic cones that reside between the flip-flops [4,5,59,63]. Social engineering is another approach, *i.e.*, an insider may provide instant access to the netlist. In any case, the goals, metrics, and processes of the first phase of hardware reverse engineering are well-defined and have vast coverage in literature. Our work mainly focuses on the second phase, *specification discovery*, which takes the gate-level netlist as an input and results in a partial or full understanding of the IC's functionality.

The reverse engineering problem lacks a formal definition and varies with each researcher's objective. Ultimately, the process shall deliver a full understanding of the entire chip or just a desired part at the architectural specification level. How to define such 'full understanding' is not exactly an engineering question. Fyrbiak *et al.* [23] discuss the human factors involved in the reverse engineering process and lay out the problem of quantifying the results. After several decades of research and practical work, this question remains unanswered.

To address that problem, we first need to answer the question of how to represent the result of the reverse engineering process. Unlike the well-structured representation at gate-level, the specification of a complex IC rather resembles a story. To enable automatic specification discovery and to allow for a quantification of success, a uniform specification format is desired. For example, raising the level of abstraction from a gate-level netlist to a high-level Register Transfer Language (RTL) style, such as Verilog or VHDL, may yield a comprehensible representation [26]. However, an unstructured RTL code lacking comments and hierarchy might still be insufficient to retrieve the specification data. Notably, automatic translation to higher abstraction level remains a hard task, hence most of the recent work relies on a library of components that serves as a reference for matching subcircuits within the investigated device [15,16,17,36,62].

In fact, this matching problem is a generalized technology mapping task. This is one of the fundamental steps in IC design synthesis and verification [39], albeit with library components on a different scale. Logic synthesis maps generic components to standard cells specified by a technology library, which comprises cells with a few input and one or two output pins. Modern CAD tools use Binary Decision Diagrams (BDD) and SAT solvers for technology mapping [6,12]. However, reverse engineering deals with large-scale circuits having tens or hundreds of pins. Testing the equivalence of two single-output functions represented as reduced, ordered BDDs can be achieved in constant time [19]. However, the size of a BDD may grow exponentially with the number of inputs. Moreover, equivalence checking requires variable correspondence between the two functions. Otherwise, all permutations must be checked, which is by itself an exponential task. In general, a Boolean matching algorithm is expected to find matches given the negation of inputs and outputs as well as input and output permutations, namely an *npnp*-invariant [14,33,44] matching. Thus, *npnp*-invariant tests are essential for hardware reverse engineering.

A modern digital IC includes millions of logic gates which are grouped into functional blocks during design. Therefore, for netlist exploration a two-phase process prevails. The first goal is to partition the netlist aiming to reconstruct the original design hierarchy. Graph-based algorithms are commonly used for circuit partitioning. The netlist can be presented as a directed graph, with logic gates being converted to vertices and nets to edges. Alternatively, both cells and nets can be represented as vertices, by that creating a bipartite graph [53]. During the first phase, this netlist graph is split into smaller subgraphs, *e.g.*, using graph density as a splitting criterion.

Next, a comprehensive library of netlist components can be used to match the resulting subcircuits. For example, syntactic matching checks for structural identity between subcircuits and library components, hence it is looking for isomorphisms. Syntactic analysis may interchangeably be referred to as structural or topological analysis. A disadvantage of the syntactic approach is its lack of flexibility. For example, the same logic function may have different implementations on the gate level [50]. In addition, structural matching will fail on even a slight deviation in the subcircuit design. Alternatively, semantic or functional analysis tests for functional identity [13]. Semantic matching is powered by formal verification methods, such as model checking using temporal logic, Boolean function property matching, or even dynamic simulation. Often, reverse engineering algorithms use a combination of different approaches. In our work we divide the existing methods into categories based on their dominant component.

Most of the available work focuses on datapath-like regular structures that operate on words [26,35, 70]. These subcircuits are easy to find thanks to their repeating patterns and can only implement a relatively limited set of functions, *e.g.*, arithmetic units, register files, funnels, and distributors. Moreover, in many circuits such structures may constitute to a majority of the overall gate count. Hence, if the success criteria is based on the sheer number of gates identified correctly, the detection of regular structures alone will generate good results. However, such criteria may be deceiving. In practice, the control circuits built of seemingly random logic are likely to contain more information despite their insignificant contribution to the gate count. How to measure the amount of information retrieved by reverse engineering is an open research question related to information theory, which, to the best of our knowledge, has not been addressed sufficiently to this date.

Despite the all-digital world, analog ICs remain an important member of the IC family. Moreover, analog ICs are a primary target for cloning and IP infringement [74]. However, cloning analog circuits is an automatic process consisting of copying the transistor-level schematics including the device sizes and electrical parameters. The commonly small footprint of analog circuits also allows for manual exploration [56].

The structure of our work broadly follows Figure 1. Section 2 briefly surveys the current state of the art in netlist extraction for ASICs and FPGAs. Section 3 gives an overview of fundamental algorithms for specification discovery from academia. Next, Section 4 describes techniques to recover the functionality from a netlist for different applications using a combination of structural and functional methods. An overview of state-of-the-art tools for hardware reverse engineering brought forward by academia and the industry is given in Section 5. Finally, Section 6 concludes our work with a discussion of open challenges and questions, as well as pointing out worthwhile future research directions.

2 Netlist Extraction

Netlist extraction aims to retrieve a human-readable netlist from an examined chip. An overview of this process is given in Figure 2. The techniques utilized for netlist extraction depend on the type of the target chip.

They fundamentally differ between fixed Application Specific Integrated Circuits (ASICs) and more flexible Field Programmable Gate Arrays (FPGAs). For both types, state-of-the-art approaches to retrieve the gate-level netlist are described in this section.

2.1 ASIC Reverse Engineering

2.1.1 Invasive Layout Recovery

The invasive reversing of an ASIC is considerably the most complex approach and gets increasingly harder due to shrinking technology sizes. The invasive netlist extraction process comprises several steps [23,31,38,58, 76], which are described in the following.

Decapsulation: the package material must be removed either by using wet or dry chemicals or by applying mechanical means. Chemicals are usually preferred since they keep the silicon die unaffected [38,58].

Delayering and Imaging: the layers of the chip are removed in interleaving with an image acquisition step to retrieve images of every layer. Since this step strongly depends on the employed manufacturing technology, a wide range of delayering techniques exists [38, 57,58]. Furthermore, one needs to differentiate between backside and frontside approaches.

Historically, frontside approaches starting at the top metal layer of the ASIC prevailed [38,58]. Here, the top passivation layer is usually removed using dry anisotropic etching due to modern feature sizes. The metal layers can then be removed by plasma etching or ion milling. This poses the challenges of over-etching, especially at the edges of the die, as well as warpages. Images of the layers may be taken using a Scanning Electron Microscope (SEM) or a Focused Ion Beam (FIB). To remove the remaining metal layers and the oxide layer, diamond suspension and dry chemistry are commonly employed. The active regions of the chip can be revealed using fluoric acid.

In recent years, however, advanced backside delayering approaches have been put forward such as automated backside thinning and plasma FIB backside delayering [57]. The latter technique even allows for the delayering and imaging steps to be performed within the same device and provides access to sub-layer information that was previously unavailable.

Processing: the images of each layer are stitched together using image processing software while ensuring precise alignment to avoid faulty transitions of adjacent images. Then, a special software is used to extract the gate-level netlist by identifying standard cells and their interconnections in the metal layers [58]. Overall, the invasive netlist extraction can only be performed by specialised labs and takes substantial time.

2.1.2 Scan-based Netlist Extraction

In contrast to the invasive extraction, a scan-based approach requires considerably less resources, albeit may be limited in accuracy. Scan insertion is a well-known Design-For-Test (DFT) technique that allows for the automatic generation of test vectors for production testing of an ASIC. Thanks to its efficiency and ability to achieve high coverage, it has become the de facto standard for testing digital circuits. The scan insertion algorithm runs at the design stage and adds to the circuit a special shift mode, which arranges all the internal registers as shift registers, so called *scan chains*. Next, it connects both sides of the chain to the chip interface. During manufacturing, the production tester uses the scan chains both to place the chip in the desired state (*ShiftIn*) and to sample its current state after operation (*ShiftOut*). These steps can be combined using a single functional (*Capture*) cycle to learn (*Probe*) the output of the combinational function F for a given input. F aggregates all the combinational logic of the chip. It receives the circuit's primary inputs and register outputs as an input vector and returns the primary outputs and register inputs as an output vector.

Scan chains provide convenient access to the IC's internal logic and can be exploited for reverse engineering. With scan-based access, heuristic algorithms can be used to find a good approximation of F, from which the learner can conjecture the circuit functionality [5].

2.2 FPGA Reverse Engineering

In contrast to ASICs, FPGAs are programmable devices that can change their functionality even after manufacturing. The FPGA's functionality is programmed using a dedicated file, the *bitstream*. For SRAM-based FPGAs, the bitstream is stored externally to the FPGA and loaded on every boot-up. The bitstream encodes the configuration of its basic logic elements as well as the interconnections between them. Hence, a bitstream is a different, usually proprietary, representation of the gate-level netlist implemented on the device.

The process of retrieving the netlist from an FPGA can be split into the steps of extracting the bitstream from memory, understanding the bitstream file format, and converting the bitstream into a gate-level netlist. To extract the bitstream, an attacker can wiretap the configuration lines on the PCB or simply read out the flash memory. The use of bitstream encryption, though not frequently used in practice, can hinder an attack.

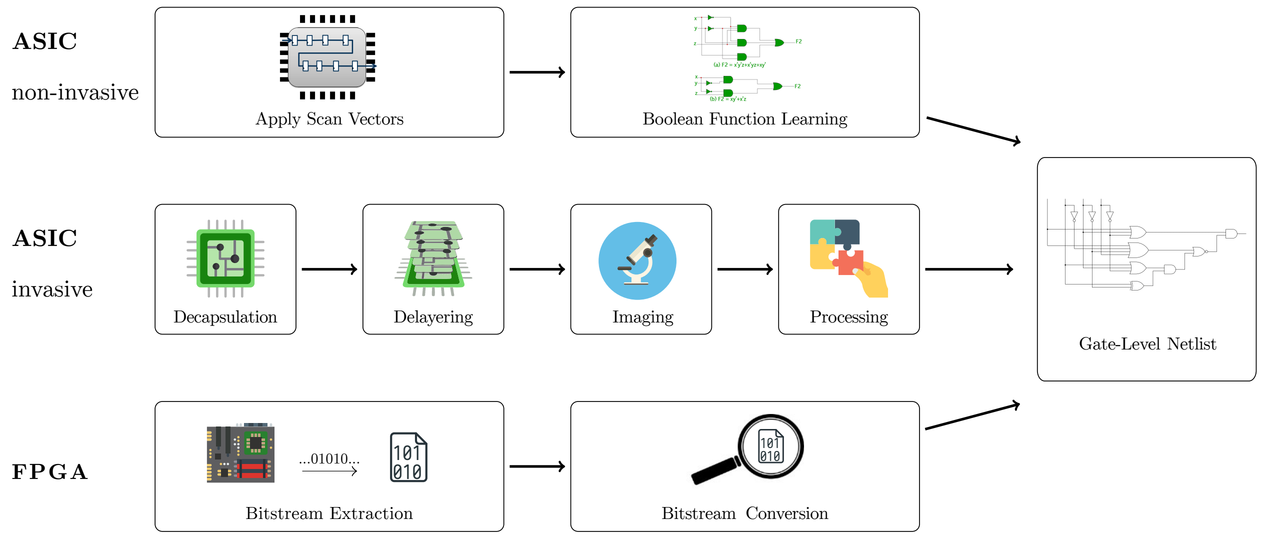

Fig. 2 Overview of the netlist extraction process for ASICs and FPGAs.

However, recent studies have shown that an attacker can overcome this obstacle using side-channel attacks [45,46,47,48,68,72] or protocol weaknesses [20].

The process of understanding the bitstream file format has recently been documented in several papers [7,28,51,55,73,79]. All these works utilize methods based on correlation, where an attacker starts with a reduced design containing only the logic element under investigation, *e.g.*, a LUT, flip-flop, or routing element. He then proceeds to vary the configuration of the element and compares the bitstreams of both designs. Since differences in the bitstream directly relate to the changes introduced in the altered design, the attacker can deduce a database that maps individual bits of the bitstream to the corresponding elements in the netlist, as well as their configuration. This database can subsequently be used by an attacker to convert the bitstream into the desired gate-level netlist.

3 Specification Discovery - Fundamental Algorithms

Netlist extraction is directly followed by the specification discovery stage. Modern techniques combine fundamental algorithms from different areas, *e.g.*, structural and functional approaches, to achieve the best results (*cf.* Section 4). This section gives some background on the historic evolution of specification discovery and hereinafter surveys the fundamental algorithms presented for this task.

3.1 Early Work

One of the first comprehensive studies describing a systematic approach for reverse engineering has been published in 1999 by Hansen *et al.* [29]. They analyze the ISCAS-85 [8] benchmark circuits available as gate-level netlists only and reveal their high-level structure.

Their analysis involves several different approaches. The `library modules` technique detects standard compound components available in the manufacturers databooks. This technique applies to circuits build in 1985, but can not be transferred to modern circuits built from high-level languages using automatic synthesis tools. `Repeated modules` searches for more instances of already identified subcircuits and thus detects regular logic. The `control functions` and `bus structures` techniques deduce additional structure and functionality by using already detected components to trace shared bus and control signals. `Common names` takes advantage of the fact that the ISCAS-85 benchmark provides the analyst with the original names of the nets.

Overall, the paper presents a set of rather intuitive semi- or fully-manual techniques that suffice for the relatively small and custom-built ISCAS-85 circuits. However, modern devices consist of millions of gates. They are being designed with the help of automated tools, which map a high level description of the design to a netlist using a library of standard cells. As a result, the netlist lacks any form of regular structure. Therefore, significantly more powerful automated tools are needed for efficient reverse engineering.

Algorithms used for Layout-versus-Schematic (LVS) verification need to combine transistors into gates to generate a gate-level netlist that can be further used for comparison with the original neltist. Application-wise, these algorithms rather belong to the netlist extraction phase. However, their underlying methods vastly resemble the ones of the specification discovery approaches presented throughout the remainder of this work. Thus, they lay the foundation for the modern approaches in reverse engineering, which is why we list them as part of the specification discovery phase. One such algorithm, SubGemini [53], identifies logic gates within a transistor-level netlist by solving a subgraph isomorphism problem. Accordingly, the circuit is first converted into a bipartite graph with both devices and nets being represented as vertices. The device vertices are labeled by their type and the net vertices by the number of connections. The algorithm then uses a label-based partitioning algorithm to solve the isomorphism between two circuits. Although the graph isomorphism problem is NP-complete, logical circuits have sufficient structure to allow for efficient solutions. SubGemini uses Breadth-first Search (BFS) with the application of hash functions. Later, Chisholm *et al.* [13] adopt SubGemini in their work to identify circuits on a higher abstraction level.

Doom *et al.* [19] propose a simulation-based method of matching subcircuits. A set of one-hot vectors are applied to the subcircuits to generate a signature from their outputs. If the subcircuits are complex enough, the presence of two subcircuits with an identical signature indicates a match with high probability. The usage of one-hot vectors makes the method permutation-invariant. This elegant heuristic is efficient, but requires a pre-processed netlist, where the subcircuits precisely matching the components are isolated in a separate cluster. However, the authors do not address the pre-processing itself.

Due to the advances in manufacturing technology that raise the scale of integrated circuits by orders of magnitude, more powerful algorithms are required.

3.2 Circuit Partitioning

Understanding circuits of millions of gates first requires partitioning into smaller subcircuits to allow for efficient understanding of high-level functionality. This step usually attempts to detect module boundaries that have been lost during synthesis. Those modules can then subsequently be investigated by the use of structural analysis as describes in Section 3.3 and functional analysis as in Section 3.4. In general, there are two approaches to tackle circuit partitioning: top-down and bottom-up.

Top-down partitioning divides a circuit into blocks using mainly graph algorithms that implement unsupervised learning. One such example are algorithms like min-cut that find a partition of a graph into subgraphs such that the number of edges between them is minimal. In [16], a pure graph-based algorithm that does not use any functional information splits the circuit into densely connected subgraphs. It employs the NCut algorithm, which is a variant of min-cut. For two subgraphs A and B, NCut is defined as the sum of weights of all the edges connecting A and B, divided by the sum of weights within each subgraph. The algorithm then finds a graph partitioning that minimizes the NCut value. Since the min-cut problem is NP-complete, a heuristic approach is used.

Graph-based top-down clustering methods are usually well-performing and require little pre-processing to deploy. However, they require good correlation between functional grouping of logic and graph density to generate accurate results. This correlation is yet to be proven. The bottom-up algorithms identify smaller subcircuits, while possibly increasing the size of the initial subcircuit by adding surrounding elements. Azriel *et al.* [4] use the Shared Nearest Neighbors (SNN) clustering algorithm to isolate stages of a cryptographic hash implementation. It operates on a flip-flop dependency graph, represented as a bipartite graph comprising flip-flop input and output groups.

Li *et al.* [35] find word-level structures in a netlist by first identifying candidates for grouping based on structural or functional similarity of wires. Next, they verify the hypothesis by forward and backward propagation, which is followed by checking whether the resulting grouping candidates still observe an identical structure.

Furthermore, Meade *et al.* [41] verify the transitive fan-in tree of cells for similarities. Cells found to be similar are then further grouped into functional blocks.

Werner *et al.* [78] apply the Louvain method for graph partitioning to iteratively optimize the quality of the partition of the network, which is also called modularity. Initially, each vertex is treated as its own cluster. Next, the method tries to merge these clusters with their neighbours based on which merging operations result in the highest modularity. Then all clusters are re-evaluated with regard to their weight. This process is repeated in an iterative manner. The authors propose to base the weight of the edges on the signal type (clock, reset, enable, select, and other), the signal distance, and its betweenness. The latter correlates to the number of shortest paths going through an edge.

Additionally, their method provides a tuning parameter γ to control the size and number of clusters, which is adapted for each case study.

3.3 Structural Analysis

Structural analysis is used to identify the high-level description of a previously extracted subcircuit by, *e.g.*, matching against a library of known subcircuits. It is confined to solely topological properties and usually pays little attention to the circuit's functionality.

Common approaches evolve around graph-based algorithms. For example, Rubanov [61] formulates the subcircuit matching problem as an optimization problem. His work is inspired by pattern recognition algorithms from the graphics domain. The investigated circuit is presented as a bipartite graph similar to the SubGemini approach described in Section 3.1. At the first stage, *discriminative labeling*, the vertices in both the objective function (the investigated circuit) and the model (library components) are labeled using a recursive labeling algorithm that considers each vertex's surroundings. Labels in the objective function consider only the vertices contained in the corresponding subcircuit. Since the boundary of the subcircuit is not known in advance, a set of labels for different radii is calculated. The second stage, *graduated assignment*, executes a non-linear optimization algorithm on the the match matrix M. This matrix contains elements for every pair of nodes in the objective graph and the model graph. If there is a match, the element value is '1', otherwise it is set to '0'. The goal is to find assignments for M that will minimize the error function. This error function reflects the matching error while taking into account the graph structure and labels. The third and final stage is the construction of subcircuit instances. The construction starts at the most probable match based on the results from the preceding stage and proceeds using BFS. Similar to SubGemini, this algorithm works for the detection of standard cells in a transistor-level netlist. It is not clear whether it is scalable enough to support larger subcircuits, because the BFS algorithm and surrounding-based labeling are prone to explode with he size of the problem.

Since netlists extracted from an IC may be incomplete and erroneous, looking for strict isomorphisms when matching with a library of subcircuits does not yield good results in practise. Instead, Fyrbiak *et al.* [25] analyze the applicability of graph similarity algorithms. Their approach uses two phases: First, they detect register stages by leveraging control signals of sequential logic elements. Next, combinational logic groups are constructed by using a reverse BFS starting at the register stages. The resulting subgraphs can then be analyzed for their similarity with components from a library. The second phase operates on the results of the first one and provides a more fine-grained investigation. Here, each subgraph is split into bitslices, which consist of Boolean functions with multiple inputs and a single output. For FPGA netlists, LUTs are decomposed into AND-OR-INV logic. Again, the bitslices are analyzed for their similarity with the candidates from the first phase. For the similarity analysis, three classifiers are investigated in detail: graph edit distance approximation, neighbour matching, and multiresolutional spectral analysis. Fyrbiak *et al.* provide a case study in the reverse engineering context, which shows promising results for circuits of a few thousand gates. However, their approach merely serves as a guideline for the human analyst in that it produces candidates that require further manual investigation.

3.4 Functional Analysis

An alternative to the structural approach is the *behavioral analysis* that examines the underlying logic functions. Again, it usually requires the netlist to be partitioned such that the investigation can be undertaken on the extracted subcircuits.

Functional analysis is heavily utilized for logic equivalence verification, which is assisted by anchors inside the circuit such as hierarchical boundaries or named sequential elements. Hence, the problem of determining equivalence is reduced to combinational matching, which can still be a hard problem requiring heuristics. For example, Agrawal *et al.* [2] propose comparing real-valued characteristic polynomials of Boolean functions. Their work shows that if the polynomials of two functions yield equal results for some group of values, the corresponding Boolean functions are equal with high probability. Computing the derivation of characteristic polynomials requires conversion of the function to a sum-of-products form, which may be a hard problem by itself. Also, in [1], Agrawal makes an interesting observation that comparing two circuits using a test-set that covers all possible faults in one of the circuits, produces correct answer with high probability. These and other combinational matching algorithms enable commercial logic equivalence verification tools working at large scale. The complexity of reverse engineering, however, is many orders of magnitude higher, since there are no anchors, like *e.g.* registers, to grab onto.

Li *et al.* [34,36] tackle the reverse engineering problem by identifying interesting behavioral patterns in the observed logic. Their method employs temporal logic to formulate the behavioral pattern. In particular, four patterns are used: *Alternating*, *Next*, *Until*, and *Eventual*. These patterns are further used as constraints in the High-Level Definition (HLD) of the circuit. The reverse engineering task then turns into a problem of matching functional blocks in the analyzed netlist with a library of pre-built HLDs. Partitioning of the design into functional blocks, a challenging problem by itself, is left out of the discussion in this paper.

One of the challenges in matching subcircuits with library components is mapping signals of the design's functional blocks to the signals in the library modules. The matching process proposed by the paper creates pattern graphs from simulations and matches the blocks by computing a maximum common subgraph, which is an NP-hard problem that is reformulated into a heuristic maximum clique problem. Afterwards follows the verification, which uses model checking [15] to verify that a candidate block satisfies the specification.

The main disadvantage of matching circuits against a library of components is the limited size of the library. Clearly, if the number of possible circuits was small enough to fit in a library, there would not be such a rich diversity of integrated circuits. Alternatively, a good way to comprehend the functionality of a circuit is back-annotating the circuit into some sort of a high-level description, *e.g.*, Verilog or VHDL. Ideally, this would be a free high-level translation. However, this task can not yet be solved by a machine. As an interim step, language templates can be used for mapping. Gascon *et al.* [26] come up with a template-based approach, where a library of templates replaces the library of components. Each template presents a generic module description that fits a broad family of modules. The core function of the family may, for example, be a counter. In this case, the template strives to include a variety of circuits that comprise a counter with some control logic. The input circuit is assumed to have passed pre-processing [70] that already partitioned it into functional blocks and identified word structures. Hence the challenge is, given a circuit with inputs comprising of words and other (control) signals, to find assignments of the control signals such that the word-level functionality of the circuit will match one of the templates. The objective is to synthesize the given circuit to a higher abstraction level that contains word-level manipulation and arithmetic operators. The matching task is represented as a Satisfiability Modulo Theory (SMT) problem and fed to the Yices SMT solver. Additional constraints help the solver to converge faster. For that purpose, the researchers use signatures based on the transitive fan-ins of the circuit outputs and transitive fan-outs of its inputs.

In [69], simulation vectors are used to deal with the sub-graph isomorphism problem. These simulation vectors are made up of one-hot and two-hot vectors that are applied to the inputs of a combinational circuit. This makes the solution permutation-invariant of the input. Using these vectors, simulation graphs are constructed. Next, sub-graph isomorphism solvers are used to locate arithmetic circuits within the combinational one. The application of this method seems to be limited to small combinational circuits only.

Observing the internals of an IC during operation by monitoring optical emissions provides additional information compared to the purely passive circuit exploration [49]. Moreover, it allows a black box analysis in case that visual access is limited due to circuit complexity or obfuscation techniques. The black-box analysis also allows to look for points of interest within the chip layout by correlating active areas with specific operations. This can greatly reduce the extend of required detailed circuit exploration.

4 Specification Discovery - Putting It All Together

A single method is not likely to suffice when dealing with real-life large-scale heterogeneous circuits. Different tools may fit different problems. Hence, for the majority of applications, a combination of algorithms from different categories are used. This section surveys the published work that combines various techniques for different purposes in IC reverse engineering.

4.1 Extracting Finite State Machines (FSMs)

Generally speaking, hardware designs can be split into datapath and control logic. The datapath is characterized by structured word-wide constructs, while the control logic lacks any obvious structure. Large parts of the IC's control logic are designed as an FSM. Hence, FSMs form an appealing target for a reverse engineer. The extraction of FSMs is a widely discussed topic in literature. It combines functional and structural methods to identify and extract the FSM circuitry and subsequently reconstruct the state graph.

4.1.1 Identification of State Registers

The first and most difficult step in the process is the correct identification of the state registers. Shi *et al.* [67] and McElvain [40] note that FSMs feature a combinational feedback path (*cf.* Figure 3). Due to their unique

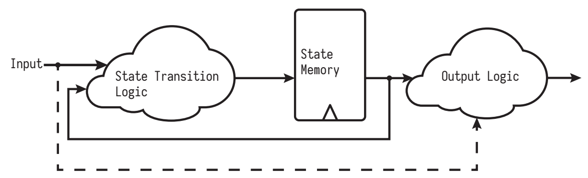

Fig. 3 Block diagram of a hardware FSM (dashed line in the case of a Mealy machine). Taken from [24].

structure, FSMs form a Strongly Connected Component (SCC) that can be identified using graph algorithms. However, this approach also marks many non-FSM circuits that form a combinational feedback path as well, *e.g.*, counters, as state machines.

Shi *et al.* [67] incorporate and extend this basic method by first selecting only registers that control datapath elements in their transitive fan-out and by ignoring the remaining ones. Identifying control signals is a challenging task on its own and is not addressed in their work. Next, registers that have the same enable signal are grouped into an individual FSM. Finally, registers that have shared gates in their combinational feedback path are grouped together. For evaluation, the authors employ the Synopsys DW8051 microcontroller and identify 36 potential state machines.

Fyrbiak *et al.* [24] improve the recognition by introducing an influence/dependence metric. Furthermore, they analyze the control behaviour of an FSM circuit candidate by determining how many gates are influenced by the FSM. Since FSMs control the data flow of the design, this metric can provide additional information to a human analyst. The decision on whether the current candidate actually represents an FSM then rests upon the reverse engineer. However, the method's performance on real-world designs remains unknown.

The RELIC tool [41] performs a topological analysis on the netlist to find similar fan-in cones of cells and is aiming to identify state registers. As a first step, the netlist is pre-processed and converted into AND-OR-INVERT logic. Next, pairs of nodes are checked for similarity by a recursive algorithm, where the graph topology serves as a similarity criterion. At each stage, the criterion is checked based on the transitive fan-in of this stage. Two nodes of the graphs are marked as similar if the value of the metric exceeds a predetermined threshold. If the gate type of the nodes differs, the traversal stops. The tool is evaluated on a number of small benchmark circuits and shows 80% to 100% accuracy in the detection of control registers. Overall, RELIC's power lays within its simplicity and it appears to be suitable for other use cases as well. However, due to its performance it seems to be applicable to small circuits only.

Brunner *et al.* [9] introduce fastRELIC improving the speed and accuracy of RELIC. It uses a grouping algorithm to reduce the number of required similarity score calculations. Their novel approach provides a speed-up of up to 100× over RELIC, which makes it more applicable to larger designs. An extensive evaluation with real-world designs ranging from 4 000 to 50 000 gates shows - with an accuracy of 23.53% to 100% - that (fast)RELIC is indeed applicable to small real-world designs.

Both RELIC as well as fastRELIC heavily rely on user-defined parameters. Thus, it is hard to tell how applicable the methods are to unknown netlists. The success rate highly varies if the parameters are not chosen correctly.

All described methods are still unreliable in the successful separation of datapath and control logic. This lies either in the nature of the methods itself or their applicability to completely unknown designs. Thus a sound identification of state registers still remains an unresolved problem.

4.1.2 Extraction of State Transitions

In contrast to the previous step, state graph extraction mostly applies functional analysis [24,40,43,42]. It starts by identifying the initial state, in which the FSM wakes up after reset. Typically, it can be deduced from either gate configuration values like, *i.e.*, initial register values (FPGAs) or the reset behaviour (ASICs).

Transitions between the states are identified by evaluating the combinational logic feeding the state registers. More precisely, each state register's data input is represented by a Boolean function, whose inputs consist of the current state, *i.e.*, the state register data output, and the external inputs to the FSM. To reveal all states that can be reached from the current state, all input combinations are applied to analyze the transition behaviour. This brute-force approach has a time complexity of $\mathcal{O}(|S| \times 2^i)$ with i being the number of external inputs to the FSM [24].

Furthermore, Fyrbiak *et al.* [24] show that most state-of-the-art FSM obfuscation schemes fail to provide the claimed level of security. The authors demonstrate how several well-established obfuscation schemes can be circumvented solely by applying the described FSM reverse engineering techniques.

4.2 Combining Structural Analysis and SAT

The formulation of the matching decision problem as a satisfiability (SAT) problem allows for the use of respective solvers, such as SAT, SMT, or CHC. These solvers manage to solve equations as large as of thousands of variables by applying heuristics. However, these heuristics only work under certain conditions, since modern solvers rely on successfully locating the internal equivalence points of the compared logic. In lack of these anchors, the solver may fail even for a small problem. The SAT solver particularly fails with different implementations of the same function, *e.g.*, different multiplier implementations.

Diao *et al.* [18] tackle the limitations of SAT solvers by converting the circuit under test to a *canonical* implementation before feeding it to the solver. For example, any multiplier detected in the circuit will be converted to the non-Booth realization type. The conversion is done using structural methods. First, basic components like half adders, full adders, and single-bit multipliers are identified. Next, their operands are identified from the resulting operator trees. The operands are then mapped using signature heuristics and the mapped structure is transformed into the canonical form, converted into a CNF, and fed to the SAT solver.

Although the scope of this paper is limited to specific arithmetic circuits, the combination of structural analysis and formal verification methods is appealing and should be further explored.

4.3 Machine Learning

Nowadays, machine learning methods of various types, especially from the area of deep learning, are penetrating many application domains. For the hardware reverse engineering problem, easily formulated as a learning problem, machine learning seems to be a natural fit. Surprisingly, very few attempts have been made by the research community to examine the usage of such algorithms. Although unsupervised learning techniques such as clustering have been used [4,16] in the past, they still fail to utilize the power of modern, advanced deep learning technologies. One possible explanation for this may simply be the lack of training data. To take advantage of deep learning, the reverse engineer has to provide huge amounts of data to train the algorithm on. This is something that appears to be impractical to achieve within the netlist domain. For example, to train a CNN to recognize an ALU in a sea of gates, it needs to be fed it with a large variety of circuits containing an ALU. However, the number of such circuits that are publicly available for analysis is insufficient for this purpose.

In one of the few initial attempts, Dai *et al.* [17] employ a CNN on the investigated circuit. The circuit is first mapped to a library built of 4-input lookup tables (4-LUTs). To make the process input permutation as well as input and output negation (npn) invariant, each 4-LUT is assigned to one out of 222 possible isomorphism classes. Every element is associated with an existence vector, which contains the classes of all the elements connected to it. The pooling layer then groups the vectors into a constant number of groups. That way, at the layer's output all the circuits have the same number of features, when k most representative vectors are taken from each group to serve as features. The resulting matrix is fed into a CNN. The network successfully solves the simple classification problem of distinguishing between multiplier and divider circuits. Furthermore, it is able to detect the presence of a multiplier or a divider in larger circuits that contain additional arithmetic units. The authors generated synthetic circuits to be used as a training set. Using a set size of 250 circuits, they achieve an accuracy of 97 to 99% for a single class, *i.e.*, deciding whether a multiplier is present in the circuit. However, the accuracy drops rapidly when increasing the number of classes. For example, using 9 classes, the accuracy drops to a range of 75 to 80%. An improvement to the authors' method is proposed in [22], mainly by compressing the existence vectors.

Chakraborty *et al.* [11] apply machine learning to attack logic locking. Such a scheme hinders reverse engineering by introducing additional logic to the design that prevents it from functioning correctly unless a secret key is supplied. The gates that mix the original logic with the key, for example XOR gates, are further obfuscated by applying an additional synthesis step. This makes the detection of the obfuscating gates by observation of the circuit practically infeasible. The authors employ machine learning to reverse the synthesis step. Notably, the model is trained with the same obfuscated circuit by applying additional random obfuscation rounds and recording changed gates and their locality.

Clearly, the current state of machine learning applications to netlist reverse engineering is not satisfactory and there is an abundant space for further exploration.

4.4 Word-Level Identification

The work by Subramanyan *et al.* [70] demonstrates how several reverse engineering algorithms from different areas can be combined to accomplish for improved performance. The paper studies the analysis of an unstructured netlist with the objective of inferring a high-level netlist with components such as register files, adders, and counters. The authors limit the scope to datapath components only, and leave the random control logic out of their consideration. They assume no access to either the RTL source code or any other microarchitectural information. Instead, only some *datasheet-level* information is available.

The authors combine structural and functional analyses into a reverse engineering process comprising two phases. First, potential module boundaries are identified using topological analysis. Finally, functional analysis is used to find potential modules and understand their behavior. They start off with a set of algorithms that identify combinational logic.

Stage 1 - bitslice identification: This stage is composed of three sub-stages: (1) All *6-feasible cuts* are enumerated. A k-feasible cut for a node N is a set of at most k nodes such that an arbitrary assignment of values to the nodes in the set completely determines the value of N [12]. The number 6 is chosen since for $k > 6$ the number of cuts increases rapidly. (2) The cuts are then grouped into equivalence classes using permutation-independent Boolean matching. The matching uses Boolean property-based signatures assigned to the input variables [44]. (3) Finally, the bitslices are aggregated into multi-bit components using two approaches. (a) Grouping based on common signals. This algorithm detects multiplexers and decoders, but also outputs candidates that can be inspected by an analyst. (b) Grouping based on a serial connection, such as in a carry propagation logic. Figure 4 illustrates the two approaches.

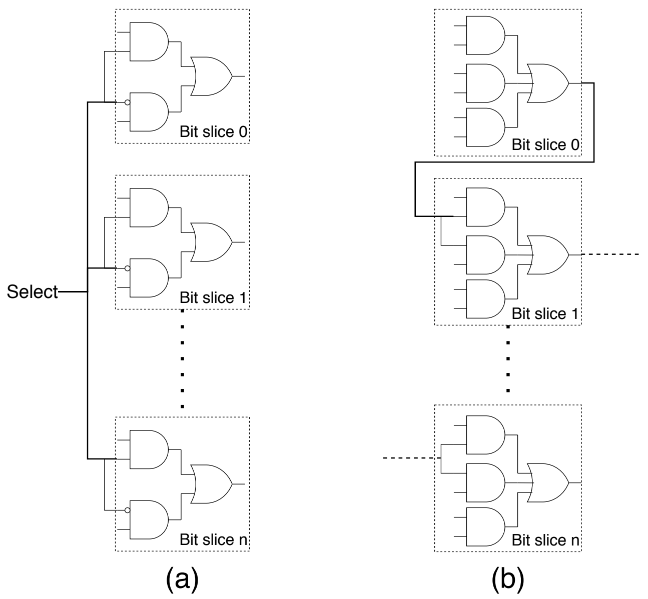

Fig. 4 Two examples of bitslice aggregation. (a) parallel connection of a select signal to bus multiplexer bit cells; (b) serial connection of a carry propagation logic.

Stage 2 - word identification: First, words are identified by grouping inputs and outputs of the aggregated slices from stage 1. Next, additional words are identified by looking at their fan-out and finding assignments to control signals that propagate a word to another candidate word.

Stage 3 - library matching: This stage identifies datapath-type modules by looking at what happens between previously identified words. Therefore, combinational blocks that have word inputs and outputs as well as side inputs are matched with reference library modules. The goal to find an assignment on the side inputs such that the function of the design block matches the function of the reference module. The equivalence problem is mapped to a 2QBF satisfiability question and a QBF solver is applied.

Stage 4 - common support: Here, nodes that have a common support are grouped together. These modules can then be verified for certain properties using BDD-based formulation. For example, to detect a pure decoder, a one-hot output property can be checked.

Stage 5 - module fusion: Related inferred modules can be fused together generate larger modules. A module fusion graph is constructed for this purpose with nodes representing modules. An edge exists only if the output of one module are inputs to the other one.

In addition to the combinational analysis, the paper also deals with sequential circuits. The analysis of sequential circuits uses a Latch Connection Graph (LCG) that is identical to the dependency graph in [4]. For example, counters have a very distinctive dependency pattern, where every consecutive node depends on all the preceding nodes. Thus, at the first stage, subgraphs that follow this rule are identified using topological analysis. Next, the candidate subgraphs are verified against the functional properties of a counter using a SAT solver. Similarly, shift registers can be detected.

RAMs and register files are identified by finding tree-type subgraphs with every node having a single fan-out. This suggest a RAM read path. BDDs are then used to verify RAM properties, *e.g.*, that each select input propagates exactly one latch to the output. Consequently, multi-bit registers are identified using the bit slice aggregation algorithm by combining a set of registers that have a common write control.

The identification of modules within the netlist may result in overlaps, which are removed at the next stage. The overlap problem is formulated as a constrained optimization problem solved using Binary Integer Linear Programming (BILP). It eventually allows for rebuilding the modules that overlap with others to remove the overlapping gates.

The algorithms presented in the paper are evaluated using eight designs mainly taken from OpenCores[1] as

[1] `https://opencores.org`

well as with a large SoC design, yielding coverage of 45 to 94%. Here, coverage describes percentage of gates that became part of one of the inferred modules. Finally, the paper additionally presents a case-study on the detection of hardware Trojans using the presented reverse engineering methods.

4.5 High-Level Register Reconstruction and Dataflow Analysis

Albartus *et al.* recently published `DANA` [3], a completely automated method to recover high-level register from completely flattened and unstructured netlists. With the help of DANA, dataflow graphs can be created to visualize the flow of data between registers. Under its hood, DANA combines various independent metrics based on structural and control information utilizing a powerful automated architecture. Notably, DANA works without any *magic* values.

`DANA` provides two modes — a *normal* and a *steered* one. In both of them the process is fully automated with the only difference being that in the steered mode a priori knowledge is applied. Hence, the reverse engineer can advise `DANA` what register sizes to expect during analysis. This information can, *e.g.*, be obtained from data-sheets or marketing materials.

Preprocessing phase: an abstracted version of the netlist is created and all combinational elements are removed. All that is left is a flip-flop (FF) dependency graph, since only the connections in between FFs are analysed during processing.

Processing phase: each of the metrics applied in this phase has to follow a fixed set of rules before a register can be created: (1) FFs need to share common clock and control signals and (2) be in the same register stage. In total, `DANA` combines nine metrics that process structural and control information while abiding to said rules. These metrics are combined with each other to find sets of registers candidates.

Evaluation phase: a *specialized majority voting* decides upon the final registers. Normal majority voting would count the number of occurrences of each register and output the register with the most votes. Meanwhile, all register candidates containing any FF of the output register is removed from the set of candidates. This has two disadvantages: (1) a priori knowledge cannot be considered and (2) small registers having lots of votes might result in a fragmentation of large registers. The authors argue that data usually flows through large registers, which are thus to be preferred. The specialized majority voting takes this into account by giving priority to registers that match a priori knowledge. Furthermore, a scan technique is implemented that analyses consequences that the selection of a register has on the others. The scan technique prefers registers that prevent fragmentation.

`DANA` comes with its own benchmark suite of nine modern hardware designs for both FPGAs and ASICs. It is evaluated using the Normalized Mutual Information (NMI) score — a statistical measure used in the evaluation of clusters. By comparing the output to the ground truth created from the synthesis report they show an almost perfect recovery of registers.

5 Reverse Engineering Tools and Frameworks

In recent years, a number of tools from academia and commercial industries have been developed to aid in the reverse engineering process. The most promising ones are presented in this section.

5.1 Academic Tools

The Hardware Analyzer (**HAL**) [10, 24, 77] is a comprehensive reverse engineering and manipulation framework for gate-level netlist. It represents the first comprehensive tool from the academic sector. HAL is designed to aid in the extraction of high-level information from gate-level netlists. The user is assumed to have no a-priori knowledge about the design hierarchy, components or synthesis tools.

HAL is not a tool by itself, but a framework to create tools. When presented with a gate-level netlist, the included HDL parsers for VHDL and Verilog convert the netlist into a directed multi-graph representation. HAL additionally allows for modularization within both, the netlist and the graph representation. The underlying gate library is read from a corresponding Liberty file by a dedicated parser. This gives HAL access to the Boolean functions implemented by each of the gates, as well as additional functionalities. For C++ and Python plugin support, HAL comes with a powerful API connected with its core library functions. A series of high-level graph algorithms is provided through an iGraph interface. However, additional algorithms may be implemented by utilizing the provided plugin system and the API. Additional features include a dedicated GUI to visually represent and interact with parts of a netlist, a Python shell, and an advanced logging system.

In [24], two offensive case studies employing HAL are presented. The first case study locates comparators used for AES self-tests by checking specific gate properties and combining the connected gates. The second case study inserts a hardware Trojan after the detection of S-boxes using the method presented in [71].

An additional tool has been brought forward by the **Degate** project [65]. It is an open IC reverse engineering framework that was developed as part of a diploma thesis. The tool receives images of the layers of an IC as well as a standard cell library and subsequently outputs a gate-level netlist. Degate offers a GUI and an API and allows for the manual grouping of standard cells into modules. Beyond that, Degate does not offer any support for automatic netlist exploration.

5.2 Commercial Tools

ChipWorks, which was recently incorporated by the Canadian company **TechInsights**, is one of the leading hardware reverse engineering service providers. Although they do only provide limited access to their tool **CircuitVision$^{\text{TM}}$** and its predecessor **ICWorks**, one can draw conclusions about their capabilities from the technical reports and papers they publish. For example, [76] shows a comprehensive analysis of the state of the reverse engineering domain back in 2009. Major parts of the paper describe the process of invasive reversing methods including process analysis and circuit extraction. They furthermore present their skill-set for package removal, delayering, imaging, and annotation. This is followed by the analysis of an extracted netlist, which is at least partly automated by *ICWorks*.

Additionally, the paper presents a case study analyzing a digital ASIC containing 12 000 gates and an EEPROM. In this study, they were able to identify register groups, main buses, and the scan path circuitry. The paper does not provide details on the methods used to identify the logic inside the chip and whether automated algorithms have been used for identification.

Similarly to ChipWorks, **Texplained** is a commercial company located in France that is selling their reverse engineering services across the industry. They develop an automatic reverse engineering tool called **ChipJuice**. This tool processes layer images of ICs, identifies wires and devices, assembles them to standard cells, and finally generates the corresponding gate-level netlist [75]. In contrast to ChipWorks, Texplained makes their tool available for use by the customer. The tool, however, does not yet provide means for the process of specification recovery.

To summarize, there is a limited tool support in throughout the industry and in academia for IC reverse engineering. Moreover, the few tools that exist are mainly dealing with the first phase of the reversing process, netlist extraction.

6 Summary, Challenges, Open Questions, and Future Directions

Hardware security in general and hardware reverse engineering in particular are in a transition stage these days. New advanced process technologies enable manufacturers to pack many more gates on a die than ever before. This additionally results in less energy being needed in order to perform arithmetic operations. Many of the traditional black-box side channel attack and fault injection techniques are not efficient enough anymore. Thus, the key to perform such attacks successfully is to use *internal information* that can be obtained by extracting the inner structure of the chip using hardware reverse engineering. The traditional approach of invasive netlist extraction is becoming more and more challenging due to the rapid advances in technology. Alternative non-invasive methods such as scan-based netlist extraction may offer a feasible alternative. As for the specification discovery stage, both the state of research and the infrastructure are still at an elementary level. We can assume that, due to the special interests of the defense sector, some developments are being undertaken behind the scenes. Hence, we can only base our judgement on openly available research.

Table 1 summarizes notable research concerning the step of specification discovery from a gate-level netlist. As one can see, comparing netlist reverse engineering methods appears to be a difficult if not infeasible task. This mainly is for two reasons. The first is the **lack of contemporary benchmark suites** that are available for evaluation. The ISCAS-85 benchmark set consists of only combinational circuits, with the largest circuit containing no more than around $3,500$ gates. Nevertheless, these 35 year-old benchmarks keep their title as being the most popular choice in hardware reverse engineering research. Results that are based on these benchmarks cannot be reliably extrapolated to the modern billion-gate SoCs. Some papers evaluate their techniques on more realistic netlists, but often fail to provide these netlists (or even just the high-level description) to the public. If someone would try to improve upon their techniques, they would not be able to compare the results in a meaningful way. Recently, Albartus *et al.* addressed this shortcoming with the benchmarks[2] published alongside `DANA` [3].

The other reason is the **lack of uniform evaluation techniques**. In 2018, Meade *et al.* [42] brought to attention that most gate-level netlist reverse engineering techniques lack proper evaluation. They point out the weaknesses of the gate coverage metrics used by some papers. In some cases, the chosen metric sim-

[2] `https://github.com/emsec/hal-benchmarks`

Table 1 Summary of specification discovery methods in IC reverse engineering. For each method, the table lists whether it utilizes structural, functional, or both approaches and whether it identifies datapath or control logic. Additionally, the gate count of the largest benchmark used for evaluation and the reported accuracy are given whenever available. Since each work solves a different problem and uses a different metric to measure accuracy, these numbers should not be used for comparison.

Publication	Year	Structural	Functional	Data	Control	Size	Accuracy	Method
Hansen [29]	1999	x	x	x	x	3 500	–	extract high-level description using manual methods
Shi [67]	2010	x			x	5 330	–	extract FSMs by detecting combinational feedback
Li [36,34]	2012	x	x	x		–	–	matching against library using behavioral patterns
Nedospasov [49]	2012		x	x		–	–	identify functional elements by correlating executed code with optical emission images
Gascon [26]	2014		x	x		3 500	–	extract high-level description using template-based subcircuit matching with SMT solver
Subramanyan [70]	2014	x	x	x		17 388	45 – 94%	combination of structural and functional methods to infer functional modules
Couch [16]	2016	x		x	x	14 972	–	split into densely connected subgraphs using NCut
Diao [18]	2016		x	x		–	93%	use SAT solver on a canonical implementation
Meade [41]	2016	x		x	x	12 576	80 – 100%	detect state registers by analyzing fan-in
Soeken [69]	2016		x	x		3 500	–	matching library components using simulation graph solving subgraph isomorphisms and SAT
Dai [17]	2017	x		x		–	99%	use CNN to classify subcircuits
Werner [78]	2018	x		x		461 511	84 – 97%	graph partitioning using Louvain method
Fyrbiak [24]	2018	x	x		x	–	–	extract FSMs by detecting SCCs, identifying feedback paths, and state transition analysis
Brunner [9]	2019	x		x	x	57 835	23 – 100%	same as [41], but improved performance
Fyrbiak [25]	2019	x		x		7 056	–	subcircuit matching using graph similarity
Albartus [3]	2020	x		x		144 303	80 – 100%	recovery of high-level registers

ply reflects how many gates were assigned into groups by an algorithm, while completely disregarding the correctness of the assignment itself.

Hence, one of the research questions that must be addressed is quantifying the success criteria of the reversing process. This metric is essential to judge and compare the proposed techniques. However, some of the metrics that researchers use to demonstrate the performance of their algorithms may be misleading. For example, measuring the percentage of gates that have been classified as part of a matched library component may miss the target of assessing the amount of information obtained during the process. However, information is what the analyst is looking for during reverse engineering. When the specification is considered, spec units, such as lines of text, can serve as information units. For example, a line that says "*the processor has a 32-bit ALU*" is one unit of information. In contrast, a complex communication protocol may need hundreds of lines to describe. This can also be normalized by the value of information.

Meade *et al.* suggest using the information-theoretic Normalized Mutual Information (NMI) as a widely accepted measurement for evaluating netlist partitioning methods. Simplified, the NMI is computed by comparing the output to a golden model. The closer the NMI is to 0, the worse is the coverage. An NMI of 1 indicates a perfect match. Implying that this leads to more unbiased results, they evaluate some of the proposed techniques and come to the conclusion that "*truthfully, the results show that there is a need for more accurate methods*" [42]. The NMI metric is a good candidate to adopt for at least some reverse engineering methods, such as netlist partitioning and separation of datapath and control logic.

As of now, there are no common tools, methods or standards in IC reverse engineering, which hinders the research progress. Coming up with a uniform infrastructure that includes intermediate representations, tools and metrics may potentially give rise to the research area as a whole. The open-source tool HAL presents a step in the right direction, although it only addresses parts of the problem.

Most of the structural methods presented for reverse engineering so far originate from the field of VLSI design and verification. This was the best fit for many years based on the size of the problem and the state-of-the-art of the algorithms. Continuous growth of the size of integrated circuits compels to explore modern, more powerful algorithms. In the recent years, graph processing algorithms gained high attention due to the growing demand in extracting information from huge graph databases, such as social networks. Allowing an approximate match, *i.e.*, a match with noise, may improve the success rate and increase efficiency [64].

In functional analysis, new heuristics should be explored to allow for faster recognition of high-level components in the netlist. So far, only very few such heuristics have been analyzed. Although the Boolean function support, also called transitive fan-in, has been proposed as a property for signatures, many more function properties can be evaluated. For example, Boolean function analysis [52] is a powerful tool that supplies properties based on influence or the Fourier analysis, such as the Walsh spectrum. Even simple Boolean properties such as symmetry, *i.e.*, input permutation invariance, or linearity can serve as good estimators. These heuristics are particularly helpful in case only partial information of the netlist is available to the analyst. In general, the case where some information on the explored netlist or the specification is available, presents a particular interest for forensic applications, such as IP theft detection and discovery of hardware Trojans.

Furthermore, the power of machine learning has a great potential to increase the efficiency of the reverse engineering process and take automation to a whole new level. However, unlike classical problems that show promising results using machine learning algorithms, such as pattern recognition in images, the size of the training dataset is strongly limited in the case of netlist reverse engineering. In the coming years, researchers should work towards overcoming this barrier. For example, the size of dataset may be increased by synthetically creating many different circuit samples using a transfer learning model [54], Generative Adversarial Networks (GANs) [27] or one-shot learning [37].

Reverse engineering does also pose a threat to IC designers and manufacturers. Therefore, companies have a legitimate desire to secure their product against reverse engineering attempts. The classification of existing work on algorithmic methods for IC reversing presented in this paper may serve as a framework for evaluation of countermeasures against different kinds of reverse engineering attacks. For example, picking different implementations of the same logical function may help to fight structural analysis attacks, since it hinders the recognition of repeating structures. However, it can not stand ground against functional algorithms that do not care for the topology of an implementation but rather for the purpose it serves. On the functional side, IC camouflaging [66] can present an efficient protection, but fails when being confronted with structural analysis. Logic locking, as first introduced in [60], is another widely studied approach to obfuscate a gate-level netlist. Engels *et al.* [21] summarize some of the existing logic locking schemes and point out their shortcomings in regard to the underlying attacker model. We see the development of effective defensive measures as one of the key challenges for future research.

Acknowledgements Part of this work was supported by the European Research Council (ERC) under the European Union's Horizon 2020 Research and Innovation programme (ERC Advanced Grant No. 695022 (EPoCH)), as well as the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA – 390781972.

References

1. Agrawal, V.D.: Choice of tests for logic verification and equivalence checking and the use of fault simulation. In: Proceedings of the IEEE International Conference on VLSI Design, pp. 306–311. IEEE (2000)
2. Agrawal, V.D., Lee, D.: Characteristic polynomial method for verification and test of combinational circuits. In: Proceedings of the IEEE International Conference on VLSI Design, pp. 341–342. IEEE (1996)
3. Albartus, N., Hoffmann, M., Temme, S., Azriel, L., Paar, C.: DANA — Universal Dataflow Analysis for Gate-Level Netlist Reverse Engineering. IACR Transactions on Cryptographic Hardware and Embedded Systems **2020**(4), 309–336 (2020)
4. Azriel, L., Ginosar, R., Gueron, S., Mendelson, A.: Using Scan Side Channel to Detect IP Theft. IEEE Transactions on Very Large Scale Integration (VLSI) Systems **25**(12), 3268–3280 (2017)
5. Azriel, L., Ginosar, R., Mendelson, A.: Revealing On-chip Proprietary Security Functions with Scan Side Channel Based Reverse Engineering. In: Proceedings of the 27th Edition o f the Great Lakes Symposium on VLSI, vol. Part F1277 (2017)
6. Benini, L., De Micheli, G.: A survey of Boolean matching techniques for library binding. ACM Transactions on Design Automation of Electronic Systems **2**(3), 193–226 (1997)
7. Benz, F., Seffrin, A., Huss, S.A.: Bil: A tool-chain for bitstream reverse-engineering. In: 22nd International Conference on Field Programmable Logic and Applications (FPL), pp. 735–738. IEEE (2012)
8. Brglez, F., Fujiwara, H.: A neutral netlist of 10 combinatorial benchmark circuits and a target translator in FORTRAN. Int. Symposium on Circuits and Systems, Special

Session on ATPG and Fault Simulation, June 1985 pp. 663–698 (1985)

9. Brunner, M., Baehr, J., Sigl, G.: Improving on state register identification in sequential hardware reverse engineering. Proceedings of the 2019 IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2019 pp. 151–160 (2019)
10. Chair for Embedded Security: HAL - The Hardware Analyzer (2019). URL `https://github.com/emsec/hal`
11. Chakraborty, P., Cruz, J., Bhunia, S.: SAIL: Machine learning guided structural analysis attack on hardware obfuscation. In: Proceedings of the 2018 Asian Hardware Oriented Security and Trust Symposium, AsianHOST 2018, pp. 56–61. Institute of Electrical and Electronics Engineers Inc. (2019)
12. Chatterjee, S., Mishchenko, A., Brayton, R., Wang, X., Kam, T.: Reducing structural bias in technology mapping. In: ICCAD-2005. IEEE/ACM International Conference on Computer-Aided Design, 2005., pp. 519–526. IEEE
13. Chisholm, G., Eckmann, S., Lain, C., Veroff, R.: Understanding integrated circuits. IEEE Design & Test of Computers **16**(2), 26–37 (1999)
14. Clarke, E., Mcmillan, K., Zhao, X., Fujita, M., Yang, J.: Spectral Transforms for Large Boolean Functions with Applications to Technology Mapping. Formal Methods in System Design **10**(2/3), 137–148 (1997)
15. Clarke, E.M., Grumberg, O., Peled, D.A.: Model checking. MIT Press (1999)
16. Couch, J., Reilly, E., Schuyler, M., Barrett, B.: Functional block identification in circuit design recovery. In: 2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pp. 75–78. IEEE (2016)
17. Dai, Y.Y., Braytont, R.K.: Circuit recognition with deep learning. In: 2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pp. 162–162. IEEE (2017)
18. Diao, Y., Wei, X., Lam, T.K., Wu, Y.L.: Coupling reverse engineering and SAT to tackle NP-complete arithmetic circuitry verification in o(number of gates). In: Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC, vol. 25-28-Janu, pp. 139–146. Institute of Electrical and Electronics Engineers Inc. (2016)
19. Doom, T., White, J., Wojcik, A., Chisholm, G.: Identifying high-level components in combinational circuits. Proceedings of the IEEE Great Lakes Symposium on VLSI (November), 313–318 (1998)
20. Ender, M., Moradi, A., Paar, C.: The Unpatchable Silicon: A Full Break of the Bitstream Encryption of Xilinx 7-Series FPGAs. 29th USENIX Security Symposium (USENIX Security 20) (2020)
21. Engels, S., Hoffmann, M., Paar, C.: The End of Logic Locking? A Critical View on the Security of Logic Locking. Cryptology ePrint Archive (Report 2019/796), 1–16 (2019)
22. Fayyazi, A., Shababi, S., Nuzzo, P., Nazarian, S., Pedram, M.: Deep Learning-Based Circuit Recognition Using Sparse Mapping and Level-Dependent Decaying Sum Circuit Representations. In: 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 638–641. IEEE (2019)
23. Fyrbiak, M., Strauss, S., Kison, C., Wallat, S., Elson, M., Rummel, N., Paar, C.: Hardware reverse engineering: Overview and open challenges. In: 2017 IEEE 2nd International Verification and Security Workshop (IVSW), pp. 88–94. IEEE (2017)
24. Fyrbiak, M., Wallat, S., Déchelotte, J., Albartus, N., Böcker, S., Tessier, R., Paar, C.: On the Difficulty of FSM-based Hardware Obfuscation. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2018, Issu**, 293–330 (2018)
25. Fyrbiak, M., Wallat, S., Reinhard, S., Bissantz, N., Paar, C.: Graph Similarity and its Applications to Hardware Security. IEEE Transactions on Computers **69**(4), 505–519 (2019)
26. Gascon, A., Subramanyan, P., Dutertre, B., Tiwari, A., Jovanovic, D., Malik, S.: Template-based circuit understanding. In: 2014 Formal Methods in Computer-Aided Design (FMCAD), pp. 83–90. IEEE (2014)
27. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems, pp. 2672–2680 (2014)
28. Guccione, S., Levi, D., Sundararajan, P., Jose, S.: JBits: A Java-based Interface for Reconfigurable Computing. 2nd Annual Military and Aerospace Applications of Programmable Devices and Technologies Conference (MAPLD) **95124**, 253–261 (1999)
29. Hansen, M., Yalcin, H., Hayes, J.: Unveiling the ISCAS-85 benchmarks: a case study in reverse engineering. IEEE Design & Test of Computers **16**(3), 72–80 (1999)
30. Kasch, S.P.: The Semiconductor Chip Protection Act: Past, Present, and Future. High Technology Law Journal **7**, 71–105 (1992)
31. Kömmerling, O., Kuhn, M.G.: Design Principles for Tamper-Resistant Smartcard Processors. Proceedings of the USENIX Workshop on Smartcard Technology (Smartcard '99) pp. 9–20 (1999)
32. Kumagai, J.: Chip detectives [reverse engineering]. IEEE Spectrum **37**(11), 43–48 (2000)
33. Lai, Y.T., Sastry, S., Pedram, M.: Boolean matching using binary decision diagrams with applications to logic synthesis and verification. In: Proceedings 1992 IEEE International Conference on Computer Design: VLSI in Computers & Processors, pp. 452–458. IEEE Comput. Soc. Press (1992)
34. Li, W.: Formal Methods for Reverse Engineering Gate-Level Netlists. Ph.D. thesis, University of California at Berkeley (2013)
35. Li, W., Gascon, A., Subramanyan, P., Tan, W.Y., Tiwari, A., Malik, S., Shankar, N., Seshia, S.A.: WordRev: Finding word-level structures in a sea of bit-level gates. In: Proceedings of the 2013 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2013, pp. 67–74. IEEE (2013)
36. Li, W., Wasson, Z., Seshia, S.A.: Reverse engineering circuits using behavioral pattern mining. Proceedings of the 2012 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2012 pp. 83–88 (2012)
37. Li Fei-Fei, Fergus, R., Perona, P.: One-shot learning of object categories. IEEE Transactions on Pattern Analysis and Machine Intelligence **28**(4), 594–611 (2006)
38. Lippmann, B., Werner, M., Unverricht, N., Singla, A., Egger, P., Dübotzky, A., Rasche, M., Kellermann, O., Gieser, H., Graeb, H.: Integrated flow for reverse engineering of nanoscale technologies. Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC pp. 82–89 (2019)
39. Mailhot, Frederic: Technology Mapping for VLSI Circuits Exploiting Boolean Properties and Operations. Ph.D. thesis, Stanford (1994)
40. McElvain, K.S.: Methods and apparatuses for automatic extraction of finite state machines (2001)

41. Meade, T., Jin, Y., Tehranipoor, M., Zhang, S.: Gate-level netlist reverse engineering for hardware security: Control logic register identification. In: 2016 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1334–1337. IEEE (2016)
42. Meade, T., Shamsi, K., Le, T., Di, J., Zhang, S., Jin, Y.: The Old Frontier of Reverse Engineering: Netlist Partitioning. Journal of Hardware and Systems Security **2**(3), 201–213 (2018)
43. Meade, T., Zhang, S., Jin, Y.: Netlist reverse engineering for high-level functionality reconstruction. Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC **25-28-Janu**, 655–660 (2016)
44. Mohnke, J., Malik, S.: Permutation and phase independent Boolean comparison. In: 1993 European Conference on Design Automation with the European Event in ASIC Design, pp. 86–92. IEEE Computer Society Press (1993)
45. Moradi, A., Barenghi, A., Kasper, T., Paar, C.: On the vulnerability of FPGA bitstream encryption against power analysis attacks: Extracting keys from Xilinx Virtex-II FPGAs. Proceedings of the ACM Conference on Computer and Communications Security pp. 111–123 (2011)
46. Moradi, A., Kasper, M., Paar, C.: Black-Box Side-Channel Attacks Highlight the Importance of Countermeasures pp. 1–18 (2012)
47. Moradi, A., Oswald, D., Paar, C., Swierczynski, P.: Side-channel attacks on the bitstream encryption mechanism of Altera Stratix II: Facilitating black-box analysis using software reverse-engineering. ACM/SIGDA International Symposium on Field Programmable Gate Arrays - FPGA pp. 91–99 (2013)
48. Moradi, A., Schneider, T.: Improved side-channel analysis attacks on xilinx bitstream encryption of 5, 6, and 7 series. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **9689**, 71–87 (2016)
49. Nedospasov, D., Seifert, J.P., Schlosser, A., Orlic, S.: Functional integrated circuit analysis. In: 2012 IEEE International Symposium on Hardware-Oriented Security and Trust, pp. 102–107. IEEE (2012)
50. Nohl, K., Evans, D., Starbug, S., Plötz, H.: Reverse-Engineering a Cryptographic RFID Tag. In: Proceedings of the 17th USENIX Security Symposium, pp. 185–194. USENIX Association (2008)
51. Note, J.B., Rannaud, É.: From the bitstream to the netlist. In: 16th International Symposium on Field Programmable Gate Arrays (FPGA), p. 264. ACM (2008)
52. O'Donnell, R.: Analysis of boolean functions. Cambridge University Press (2014)
53. Ohlrich, M., Ebeling, C., Ginting, E., Sather, L.: SubGemini: identifying subcircuits using a fast subgraph isomorphism algorithm. In: Proceedings of the 30th international on Design automation conference - DAC '93, pp. 31–37. ACM Press, New York, New York, USA (1993)
54. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering **22**(10), 1345–1359 (2010)
55. Pham, K.D., Horta, E., Koch, D.: BITMAN: A tool and API for FPGA bitstream manipulations. In: Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 894–897. IEEE (2017)
56. Polian, I.: Security aspects of analog and mixed-signal circuits. In: 2016 IEEE 21st International Mixed-Signal Testing Workshop, IMSTW 2016. Institute of Electrical and Electronics Engineers Inc. (2016)
57. Principe, E.L., Asadizanjani, N., Forte, D., Tehranipoor, M., Chivas, R., DiBattista, M., Silverman, S., Marsh, M., Piche, N., Mastovich, J.: Steps toward automated deprocessing of integrated circuits. In: ISTFA 2017: Proceedings from the 43rd International Symposium for Testing and Failure Analysis, pp. 285–298. ASM International (2017)
58. Quadir, S.E., Chen, J., Forte, D., Asadizanjani, N., Shahbazmohamadi, S., Wang, L., Chandy, J., Tehranipoor, M.: A survey on chip to system reverse engineering. ACM Journal on Emerging Technologies in Computing Systems **13**(1) (2016)
59. Rolt, J.D., Natale, G.D., Flottes, M.L., Rouzeyre, B.: A novel differential scan attack on advanced DFT structures. ACM Transactions on Design Automation of Electronic Systems **18**(4), 1–22 (2013)
60. Roy, J.A., Koushanfar, F., Markov, I.L.: EPIC: Ending piracy of integrated circuits. In: Proceedings of the conference on Design, automation and test in Europe, pp. 1069—-1074 (2008)
61. Rubanov, N.: SubIslands: the probabilistic match assignment algorithm for subcircuit recognition. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **22**(1), 26–38 (2003)
62. Rubanov, N.: A High-Performance Subcircuit Recognition Method Based on the Nonlinear Graph Optimization. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **25**(11), 2353–2363 (2006)
63. Saab, D.G., Nagubadi, V., Kocan, F., Abraham, J.: Extraction based verification method for off the shelf integrated circuits. In: 2009 1st Asia Symposium on Quality Electronic Design, pp. 396–400. IEEE (2009)
64. Samanvi, K., Sivadasan, N.: Subgraph Similarity Search in Large Graphs. arXiv (2015)
65. Schobert, M.: Interactive Functions of the Degate Software Package (2012)
66. Shakya, B., Shen, H., Tehranipoor, M., Forte, D.: Covert Gates: Protecting Integrated Circuits with Undetectable Camouflaging. tCHES 2019 **2019**(3), 86–118 (2019)
67. Shi, Y., Ting, C.W., Gwee, B.H., Ren, Y.: A highly efficient method for extracting FSMs from flattened gate-level netlist. In: Proceedings of 2010 IEEE International Symposium on Circuits and Systems, pp. 2610–2613. IEEE (2010)
68. Skorobogatov, S., Woods, C.: Breakthrough silicon scanning discovers backdoor in military chip. In: E. Prouff, P. Schaumont (eds.) Cryptographic Hardware and Embedded Systems - CHES 2012, *Lecture Notes in Computer Science*, vol. 7428, pp. 23–40. Springer, Berlin, Heidelberg (2012)
69. Soeken, M., Sterin, B., Drechsler, R., Brayton, R.: Simulation graphs for reverse engineering. Proceedings of the 15th Conference on Formal Methods in Computer-Aided Design, FMCAD 2015 pp. 152–159 (2016)
70. Subramanyan, P., Tsiskaridze, N., Li, W., Gascón, A., Tan, W.Y., Tiwari, A., Shankar, N., Seshia, S.A., Malik, S.: Reverse engineering digital circuits using structural and functional analyses. IEEE Transactions on Emerging Topics in Computing **2**(1), 63–80 (2014)
71. Swierczynski, P., Fyrbiak, M., Koppe, P., Paar, C.: FPGA Trojans Through Detecting and Weakening of Cryptographic Primitives. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **34**(8), 1236–1249 (2015)
72. Swierczynski, P., Moradi, A., Oswald, D., Paar, C.: Physical security evaluation of the bitstream encryption mechanism of altera stratix II and stratix III FPGAs. ACM

Transactions on Reconfigurable Technology and Systems **7**(4) (2014)

73. SymbiFlow: Project X-Ray (2018). URL `https://github.com/SymbiFlow/prjxray`
74. Technology, S.S.: Top 5 counterfeited semiconductors: Analog ICs top the list — Semiconductor Digest. URL `https://sst.semiconductor-digest.com/2012/04/top-5-counterfeited-semiconductors-analog-ics-top-the-list/`
75. Thomas, O., Sarl, T., Nedospasov, D.: On the Impact of Automating the IC Analysis Process. Tech. rep. (2015)
76. Torrance, R., James, D.: The state-of-the-art in IC reverse engineering. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **5747 LNCS**, 363–381 (2009)
77. Wallat, S., Albartus, N., Becker, S., Hoffmann, M., Ender, M., Fyrbiak, M., Drees, A., Maaen, S., Paar, C.: Highway to HAL: Open-Sourcing the First Extendable Gate-Level Netlist Reverse Engineering Framework. ACM International Conference on Computing Frontiers 2019, CF 2019 - Proceedings pp. 392–397 (2019)
78. Werner, M., Lippmann, B., Baehr, J., Grab, H.: Reverse engineering of cryptographic cores by structural interpretation through graph analysis. 2018 IEEE 3rd International Verification and Security Workshop, IVSW 2018 pp. 13–18 (2018)
79. Ziener, D., Aßmus, S., Teich, J.: Identifying FPGA IP-cores based on lookup table content analysis. Proceedings - 2006 International Conference on Field Programmable Logic and Applications, FPL pp. 481–486 (2006)