ReCon: From the Bitstream to Piracy Detection

Grant Skipper*, Christopher Sozio*, Adam Duncan*†, Andrew Lukefahr*, Martin Swany*

*Intelligent Systems Engineering, Indiana University, Bloomington, Indiana 47401 USA
†NAVSEA Crane, Crane, Indiana 47522 USA
Email: gskipper@iu.edu

***Abstract*—Field-Programmable Gate Arrays (FPGAs) are used almost everywhere, from smart-phones to datacenters. FPGA functionality is determined by the intellectual property (IP) encoded within a vendor-proprietary binary configuration file, or bitstream, that is deployed on these devices. The potential value offered by IP creates many incentives for an adversary to attempt to steal it. The opaque nature of vendor bitstream formats has thus far limited the ability to detect stolen IP in deployed FPGAs. In this work we demonstrate the ability to detect instances of IP piracy given ONLY the FPGA configuration bitstream and a golden user netlist.**

While prior works have shown it is possible to reverse-engineer FPGA bitstreams into an underlying structural netlist, in isolation this netlist is of limited utility. We take this a step further by introducing ReCon, a tool to automatically assist in detecting pirated IP within a bitstream. ReCon first extracts a netlist from an unknown bitstream, then applies subgraph isomorphism algorithms to detect IP within the extracted netlist that are pirated from an original, or user, IP. Our experiments demonstrate ReCon as an effective method for piracy detection because it allows for a composite comparison between two bitstreams without needing to manually reverse engineer circuits and identify submodules.

I. INTRODUCTION

The flexibility of Field-Programmable Gate Array (FPGA) devices as reprogrammable hardware makes them an attractive option for many commercial applications such as cars, smart-phones [1], and datacenters [2]. The ability to target FPGAs as modular hardware has helped a robust Intellectual Property (IP) licensing market develop around this class of device. The value of IP on modern FPGAs makes it a tempting target for piracy. In this paper we define FPGA IP piracy, or simply piracy, as attempting to extract IP from FPGAs with the goal of circumventing existing property protection laws.

A *bitstream* is a binary configuration file that is loaded into FPGA memory and includes the encoding the architecture needs to realize a user design. In most commercially available FPGAs, the bitstream format is highly opaque to users, and is often a closely-guarded vendor-proprietary secret [3]. Thus reverse-engineering the bitstream entails the ability to extract IP directly from FPGA bitstreams.

A traditional technique for detecting IP piracy is watermarking. This anti-piracy technique involves adding extra logic or traits in the netlist or register transfer level (RTL) code of an FPGA design to embed a unique feature in the IP that can be detected during later analysis [4]. However, the overhead of needing to add watermarking features bloats design complexity and can be potentially analyzed and overcome by malicious actors.

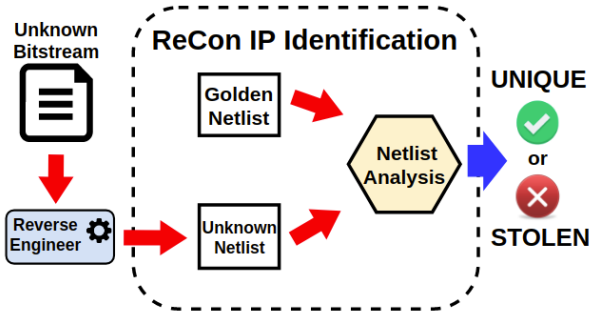

Fig. 1: ReCon compares a user netlist with a structural netlist reverse-engineered from an unknown bitstream to perform IP piracy detection.

We present an IP piracy detection algorithm, ReCon, which can detect IP piracy given only access to an FPGA bitstream or configuration memory as well as the original IP. ReCon works by first extracting a structural netlist from an unknown bitstream, and then applying a two-step subgraph isomorphism algorithm to determine the likelihood that IP piracy/theft has occurred (Fig.1).

This work makes the following contributions:

- We show it is possible to detect user IP in an unknown bitstream given only a target bitstream and the original user netlist.
- We demonstrate that by exploiting the regularity of the FPGA fabric we can apply subgraph isomorphism techniques to perform piracy detection without relying on additive watermarks or manual circuit reverse engineering.

II. BACKGROUND

Modern FPGAs have introduced multiple mechanisms for mitigating piracy such as on-chip encryption/decryption engines [5], options to disable configuration readback [6], and authentication schemes [3]. However, these integrated FPGA security features have repeatedly proven susceptible to physical attacks [7][8][9]. Software-based features such as bitstream file encryption and authentication schemes are susceptible to key extraction [10] and other attacks [11] [12] thus **cryptography cannot be relied on as the sole mechanism for mitigating theft**

*1) **Reverse Engineering**:* Researchers have explored reverse-engineering bitstreams back to a netlist for well over the past decade [13][14][15]. However, it has not been until the past few years that techniques for easily manipulating [16] and reverse engineering complex bitstreams such as in modern Xilinx devices have been published [17][18]. New strategies for detecting tampered or stolen IP must incorporate advancements in FPGA reverse-engineering techniques to keep pace with the increasing sophistication of reverse engineering tools.

*2) **Watermarking**:* Watermarking is a common defensive technique to assist in determining authenticity of IP. Watermarking techniques add identifying features that do not impact the functionality of the design. There are three commonly-used watermarking approaches: RTL-based, netlist-based, and bitstream-based - unfortunately, all add overhead, and can fail in the face of reverse-engineering-based piracy.

RTL-based watermarking, such as work demonstrated by Cui et al.[19], are only practical when the designer has access to the pirated RTL. Thus, this type of watermarking is unable to detect IP piracy when the design is only available as a bitstream.

Netlist-based watermarks attempt to embed watermarking logic such that the output of the watermark is a function of the underlying design [20][21]. While detecting a netlist watermark is strong evidence of piracy, the lack thereof does not prove originality. Reverse-engineering configuration to defeat this type of verification has been demonstrated in recent FPGA tampering attacks [22] and is prone to reversal with simple bitstream analysis tools.

Bitstream watermarking techniques that add or change configuration encoding are attractive because the unique features of the watermark are easily detected via static analysis [23]. The same tooling and techniques used to implement and verify this class of watermark, however, can be also be used to circumvent this defense and hide evidence of piracy. For this reason, piracy detection strategies must incorporate the full information as described by the bitstream rather than narrowing analysis to a handful of potential traits expressed in the binary.

III. Threat Model

In our threat model we assume an attacker may illicitly acquire IP in netlist form through a number of methods. For example, the encryption keys for proprietary IP may be compromised, an employee or affiliate may illicitly steal or send netlist IP for monetary gain [24], or the bitstream of a design may be decrypted and reverse-engineered into a netlist [17].

Our threat model also assumes that we, as analysts, are able to obtain the decrypted bitstream of a potentially pirated design. If a bitstream is encrypted we assume we can decrypt it by applying previously published techniques for physically extracting encryption keys from the device, such as thermo-laser stimulation [10]. We also assume that the analyst has access to the original user IP and this IP is distributed as a netlist - not as RTL code. Companies such as Xilinx and other 3rd party suppliers distribute their commercial IP products as fixed encrypted netlists [25]. Similar to related work on reversing-engineering digital circuits [26], we also assume an analyst has access to public "datasheet" knowledge of the device being analyzed.

IV. Piracy Detection: Obfuscation and Challenges

Detecting piracy through pure binary file analysis is difficult. First, even small changes in the design flow can introduce major changes in how the resulting bitstream is encoded. Second, a sophisticated attacker might also make small changes to the spacial layout of the pirated IP in the FPGA to deliberately change the binary footprint of the bitstream. Therefore, the goal of our ReCon technique attempts to move beyond pure binary analysis of the bitstream file itself.

Our tool must also detect piracy despite obfuscation from netlist tampering. A number of netlist tampering techniques exist for obscuring pirated IP. For example changing Look-Up-Table (LUT) encodings and LUT input pins within a stolen netlist, stuffing unused or unimportant areas of LUT contents with garbage encoding, malicious place and route changes, and adding dummy circuitry or resources into the design. As long as the functional integrity of the IP persists in an unknown bitstream we can determine with confidence whether or not user IP is present in spite of adversarial changes to the netlist.

V. ReCon IP Piracy Detection

We introduce ReCon, an algorithm that can detect pirated IP inside of an unknown bitstream without relying on manual circuit reversal. ReCon first parses the unknown bitstream into the underlying structural netlist. Next, ReCon uses recursive subgraph isomorphism techniques to compare the unknown netlist to a user IP to identify local similarities between the two. Finally, ReCon returns a threshold value of functional similarity between the two.

ReCon is meant to expedite IP piracy detection in FPGAs and simplify black-box analysis of unknown bitstreams. Our technique diverges from other digital circuit reverse engineering strategies in two distinct ways. First, by focusing our algorithm on FPGAs we can exploit the regularity of the architecture to perform composite analysis. And second, our technique is a fully automated process that does not rely on manual or incremental analysis to determine the likelihood of piracy.

Without ReCon an analyst would need to manually follow a netlist to identify potential elements of user IP, and then also separate the user IP from the larger design to confirm its presence. Current techniques for accomplishing this analysis in a non-trivial design can be a challenging and time-consuming task. With ReCon we can algorithmically perform this analysis and accurately identify the existence user IP.

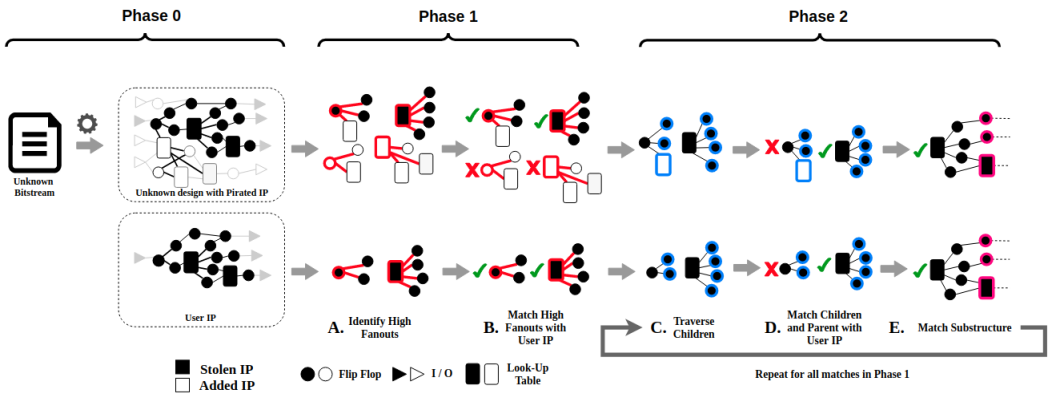

Fig. 2: Overview of ReCon algorithm. Left to Right: **A)** Gather logic and fanout metadata. **B)** Match resources in both netlists based on fanout contents. **C)** For each matching substructure, traverse downhill resources. **D)** During traversal, compute similarity of potential matching substructures. **E)** Verify if substructure is found in the user IP, and if applicable, fits with any previously processed structures. Repeat this process for all potential structural matches found in step B.

A. Phase 0: Bitstream Reverse Engineering

We employ bitstream reverse engineering strategies elaborated by Zhang et all. [17] to take an unknown bitstream and transform it back into an unknown netlist. The reverse-engineering process only requires access to 'datasheet level' knowledge of the part to accurately reverse a design. The unknown netlist produced by this reverse engineering describes connectivity between in-fabric architecture primitives such as Look-Up-Tables (LUTS), Flip-Flops (FFs), Muxes, I/O, Clocks, Block-RAM (BRAM), etc.

After reverse-engineering a bitstream back into a netlist, it can be treated as a graph. Each node in a netlist graph corresponds to a resource element in the fabric, such as a LUT or Flip Flop. Edges in a netlist graph represent signals that connect logic resource elements. We use the relationships between resource nodes and their corresponding fanout edges to identify similarities between the unknown design and the user IP.

B. Phase I: Group Fanouts & Identify Clocks

1) **Fanout Analysis:** After reverse-engineering the bitstream, ReCon identifies high-fanout logic elements within the unknown netlist and attempts to correlate these elements with corresponding high-fanout elements within the user IP. At the conclusion of this phase, we have a tractable number of potential matching sub-graphs in the netlist we can further process to determine likeness.

FPGA netlists are constrained by the types of logic resources available in the architecture. This regularity can exploited to more easily discern matching functionality between two designs and perform analysis. Typical digital logic circuits contain a small number of high-fanout nets, e.g. clock, reset, and enable signals.

ReCon starts by identifying resources between the user netlist and the extracted unknown design that share similar degrees of fanout connections [Fig.2A]. Resources with a high degree of children are more likely to contain detailed features that can be compared between the two designs. Both the number of fanout connections and the type of resources they fan out to are considered during this phase. Grouping resources by these features narrows the search space for graph traversal in the next phase [Fig.2B].

2) **Clock Analysis:** Sequential designs that use separate clock domains or implement circuits that drive the clock to sub-resources can be identified within a reverse engineered bitstream. Using the target bitstream, we follow the global clock signals as they spread through the fabric and separate nets that rely on different signals for clocking. If two or more clock signals are present in the netlist then the resources can be separated into sub-graphs based on the driving clock signal We can traverse the connections of these resources to discriminate the clock domain of downhill resources.

Each clock-based resource group found in the unknown netlist can be compared to the user netlist by analyzing the sequence of connections and children found in each group in the user and unknown netlist. Groups with resources that drive the same number and types of downhill nodes are considered 'positive' matches that we should continue to traverse their children to further develop the extent of the match.

C. Phase II: Traversal and Comparison

Logic resources gathered from each netlist graph are individually compared based on their fanout connections. While resource placement may differ between two bitstreams, the functional behavior of the netlists will match if the user IP is the same or a subset of an unknown design.

Fanout signals for each resource are traversed to identify the type and number of logic resources that have inputs driven by a common signal [Fig.2C]. Because structural netlists are less affected by optimizations made in the design flow than RTL, similar groups of resources can be observed in both netlists if subsections of the two are derived from the same source. To match resources during traversal, resources are compared in each design based on the type of resource and

the number of fanout connections associated with it in the netlist. If a resource in the original netlist has one or more corresponding resources with the same number and type of fanout connections we then process this pair further as a prospective match [Fig.2D].

Not all matching resource and fanout relationships are indicative of matching IP. Further processing must be done before we can evaluate confidence on whether or not this matching pair represents the 'same' functional element. We do this by traversing each child node in the corresponding match in the user design and cross referencing the fanout relationship of the child with the children of the matched resource in the unknown netlist [Fig.2E]. If the children of each resource are found to each have a matching resource in the other set, then this grouping of nodes can be determined to be functionally equivalent at this stage.

Sometimes a resource belonging to the original user IP may have more fanout connections expressed in the unknown design than in the original design. This may happen when specific signals are being shared between various design elements such as a design wide 'reset' or 'enable' signal being shared by both user IP and elements in the unknown design. If matching pairs are observed between both designs, we apply subgraph backtracking [27] on relevant fan-in connections to reevaluate resources being shared with user IP in an unknown netlist.

Subgraph traversal terminates when there are no unique children that can be traversed. Generally termination happens when a child node's next step is to a child of a subgraph that has already been processed, when the only untraversed step left loops back on a child node whose paths have already been processed, or when the next step of the subgraph is to an I/O resource - which indicates completion of the current path.

D. *Similarity Rating*

After processing the logic elements in both the user and unknown designs, a similarity coefficient is calculated with respect to the user netlist. We use the *Szymkiewicz–Simpson*, or overlap coefficient [28] as our similarity metric to account for the potential that either of the two designs being compared represent a subset of the other.

$$overlap(N_u, N_b) = \frac{N_u \cap N_b}{min(N_u, N_b)} \tag{1}$$

Our usage of this coefficient denotes the *ratio* of common structural similarity between an unknown design and the user IP. The intersection of user netlist N_u and unknown design N_b is composed of elements of between designs that demonstrate a high degree of structural equivalence to each other. A high similarity index reflects a high ratio of elements in a user IP are expressed in the unknown design.

When ReCon determines a high confidence that user IP is present in an unknown design, an analyst can then use the matching subgraph found by ReCon for more targeted analysis, such as isolated simulation. This contrasts to simply reversing an unknown bitstream and simulating aspects of the netlist to try and determine piracy. ReCon allows analysts to quickly identify the structure in the netlist that corresponds to stolen IP and perform further analysis steps such as focusing simulation on the matching substructure.

VI. Results and Evaluation

We tested ReCon's IP detection with netlists of different benchmark circuits as the original user IP. We "stole" the netlist by reverse-engineering a bitstream back into a netlist and integrating it with other modules to reflect illicit deployment of the IP in a larger unknown design. Then, ReCon is able to discern the functionality and structure of the user IP in malicious designs with confidence rating shown in Table 1. We also demonstrate that a low similarity index is calculated when comparing two designs with incongruous netlists.

*1) **Experimental Setup**:* Our IP piracy detection experiments were performed on a Xilinx Artix-7 xc7a35ticpg236-1L. Our experimental model uses various real-world designs from open-source projects [**?**] as protected IP that we want to detect in potentially malicious designs. During experimentation we synthesize our user circuits with other larger designs and allow the synthesis process to optimize and change resource place and routing.

*2) **Basic Identification**:* IP detection using a reverse engineered unknown netlist must be resilient to place and route permutations. We demonstrate that our IP detection method is resistant to spatial place and route changes by generating multiple bitstream designs from a user netlist for various designs with different place and route constraints. We use these bitstreams to then evaluate how accurately ReCon can detect pirated IP.

Table 1.1 describes the accuracy of finding netlists within bitstreams composing only of designs with varying placement in the fabric. The resulting similarity index between the original netlist B and B' permutations of the same circuit demonstrates a high confidence rating that the golden netlist is present in the extracted bitstream - regardless of place and route changes.

To demonstrate that ReCon can detect IP regardless of its spatial implementation we also show accuracy when user IP is not in the test bitstream. We tested for negative scenarios where a user IP is not present in a design by comparing the golden B' user netlist with bitstreams that do not include the user circuit at all, but instead another design. As seen in Table 1.3, when our algorithm cannot confidently match the golden B' netlist within a bitstream, a low confidence rating is associated with the result.

*3) **Complex Identification**:* Detecting IP when it is used as part of a larger design is complicated when the design tools closely integrate logic during synthesis and implementation. To overcome optimizations in the design flow ReCon must be resilient to these changes and able to detect IP given variations in the design flow.

For this evaluation we experimented with finding user IP in larger composite designs by taking user IP and synthesizing

TABLE I: Results of ReCon algorithm on various trials for detecting stolen IP directly from a bitstream.

User IP	Extracted Bitstream	Detected In Design	Similarity Index	Place & Route	Resources (User / Unknown)
	1) Self-IP identification.				
PicoRiscV32 [29]	PicoRiscV32	Yes	0.95	Changed	1721 / 1721
OpenRisc1200 [30]	OpenRisc1200	Yes	0.93	Changed	11258 / 11258
DarkRisc [31]	DarkRisc	Yes	0.93	Changed	1456 / 1456
	2) Pirated IP in bitstream.				
PicoRiscV32	UART + PicoRiscV32	Yes	0.92	Changed	1721 / 1996
PicoRiscV32	UART + SHA128 + PicoRiscV32	Yes	0.90	Changed	1721 / 2724
OpenRisc1200	SHA128 + OpenRisc1200	Yes	0.89	Changed	11258 / 11986
OpenRisc1200	UART + SHA128 + OpenRisc1200	Yes	0.89	Changed	11258 / 12361
	3) No Pirated IP in bitstream				
PicoRiscV32	DarkRisc	-	0.26	Changed	1721 / 1456
SHA128 [32]	PicoRiscV32	-	0.11	Changed	678 / 1721
OpenRisc1200	PicoRiscV32	-	0.23	Changed	11258 / 1721
DarkRisc	SHA128	-	0.15	Changed	1456 / 678

with other RTL modules. When synthesized in this manner the user netlist may become optimized in the fabric during place and route to ensure timing and circuit efficiency is met. Results for this evaluation are demonstrated in Table 1.2.

For this experiment we generated bitstreams of various processor core RTL modules with different modules that interfaced with the IO of each design. Each bitstream had different constraints on place and routing to coerce the design process to optimize each implementation differently. ReCon determined a high confidence that the individual user designs were contained in each bitstream, calculating at the lowest, a .89 similarity index that the golden IP was in the design.

Next we attempted to discern the user netlist in a more complex composite design. This test incorporateds other unrelated RTL modules in addition to the user netlist. The introduction of these extra elements into the bitstream tests ReCon's ability to identify false positives when reconstructing the user IP from the larger design. For ReCon to accurately identify the user IP in this bitstream it must also be able to disentangle unknown logic from the user design after optimization by the tools. The post-implementation phase of these experiments consists of highly entangled resources between these modules. ReCon analyzed a high similarity index of around .90 in this experiment. The resulting similarity index successfully determined a high-likelihood that the user netlist is included in this bitstream.

*4) **Evaluation**:* The experiments conducted in Table 1 demonstrate that detecting a user IP in a bitstream is resistant to changes in placement and routing. We implemented these experiments to test the ability of ReCon to establish baseline results and show how accurate IP detection is possible given *only* a bitstream and the original user netlist.

Based on our implementation and experiments, the similarity index of IP being when it is included in an unknown bitstream is generally found to be around .90 or higher. When an IP is not present in the unknown bitstream the similarity intersection is less than .3.

Dissimilar designs being given a similarity rating higher than 0.2 occur because non-trivial designs will share a number of basic logic structures. ReCon attempts to match all parts of the golden netlist into the unknown design. This includes attempting to traverse logical elements that have very low fanout connections. Between any given set of netlists, there is a high probability of finding many 1-to-1 connections between logic resources. For example, the pattern of one Flip Flop fanning out to a single downhill AND-gate occurs frequently in many netlists. When this pattern is present it is very likely that this will also be found in the unknown bitstream as this is a common structure in sequential circuits. In our experimental data we chose to include *all* matching substructures in ReCon's analysis, however the algorithm can be constrained to devalue common netlist structures when determining similarity.

ReCon's focus on the structural and functional constraints of in-fabric resources allows for fast comparison and identification of design similarities. ReCon uses the structure found within a bitstream to make decisions about potential matches between designs.

VII. Conclusion

In this paper, we presented the ReCon IP piracy detection algorithm to identify user IP directly from a bitstream. The ability to detect IP piracy from an unknown bitstream without the use of additive watermarking techniques represents a divergence from traditional forms of FPGA IP watermarking. When using the ReCon algorithm *golden user IP essentially becomes the watermark for its own design*. In our experiments we showed how the ReCon algorithm is resilient to changes in the bitstream that may occur due to variance in the design flow. Our experimental results of the ReCon algorithm also demonstrated accurate detection of user IP both when it is stolen as a standalone design and also when it is embedded in a larger complex design.

References

[1] A. Tilley, "This mysterious chip in the iphone 7 could be key to apple's ai push," *Forbes*, Oct 2016.

[2] S. Trimberger and S. Mcneil, "Security of fpgas in data centers," *2017 IEEE 2nd International Verification and Security Workshop (IVSW)*, 2017.

[3] S. Trimberger, J. Moore, and W. Lu, "Authenticated encryption for fpga bitstreams," in *Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays*, pp. 83–86, ACM, 2011.

[4] J. Zhang and G. Qu, "A survey on security and trust of fpga-based systems," *2014 International Conference on Field-Programmable Technology (FPT)*, pp. 147–152, 2014.

[5] K. Wilson, "Xapp1239 - using encryption to secure a 7 series fpga bitstream," 2018.

[6] E. Peterson, "Xapp1084: Developing tamper resistant designs with xilinx virtex-6 and 7 series fpgas," *Xilinx Application Note*, 2017.

[7] A. Moradi and T. Schneider, "Improved side-channel analysis attacks on xilinx bitstream encryption of 5, 6, and 7 series," in *Workshop on Constructive Side-Channel Analysis and Secure Design*, Springer, 2016.

[8] A. Moradi, D. Oswald, C. Paar, and P. Swierczynski, "Side-channel attacks on the bitstream encryption mechanism of altera stratix ii: facilitating black-box analysis using software reverse-engineering," in *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays.*, pp. 91–100, ACM, 2013.

[9] R. Karam, T. Hoque, S. Ray, M. Tehranipoor, and S. Bhunia, "Power analysis attack on hardware implementation of mac-keccak on fpgas," in *ReConFigurable Computing and FPGAs (ReConFig), 2016 International Conference on*, pp. 1–8, IEEE, 2016.

[10] H. Lohrke, S. Tajik, T. Krachenfels, C. Boit, and J.-P. Seifert, "Key extraction using thermal laser stimulation: A case study on xilinx ultrascale fpgas," in *IACR Transactions on Cryptographic Hardware and Embedded Systems*, IACR, 2018.

[11] M. Ender, A. Moradi, and C. Paar, "The unpatchable silicon: A full break of the bitstream encryption of xilinx 7-series fpgas," in *29th USENIX Security Symposium (USENIX Security 20)*, (Boston, MA), USENIX Association, Aug. 2020.

[12] A. Moradi, A. Barenghi, T. Kasper, and C. Paar, "On the vulnerability of fpga bitstream encryption against power analysis attacks," *Proceedings of the 18th ACM conference on Computer and communications security - CCS 11*, 2011.

[13] E. R. Jean-Baptiste Note, "From the bitstream to the netlist," in *Proceedings of the 16th International ACM/SIGDA Symposium on Field Programmable Gate Arrays*, FPGA '08, (New York, NY, USA), pp. 264–264, ACM, 2008.

[14] S. A. H. Florian Benz, André Seffrin, ""bil: A tool-chain for bitstream reverse-engineering"," in *22nd International Conference on Field Programmable Logic and Applications (FPL)*, pp. 29–31, 2012.

[15] Z. Ding, Q. Wu, Y. Zhang, and L. Zhu, "Deriving an ncd file from an fpga bitstream: Methodology, architecture and evaluation," *Microprocessors and Microsystems*, vol. 37, no. 3, pp. 299–312, 2013.

[16] K. D. Pham, E. Horta, and D. Koch, "Bitman: A tool and api for fpga bitstream manipulations," in *2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 894–897, IEEE, 2017.

[17] T. Zhang, J. Wang, S. Guo, and Z. Chen, "A comprehensive fpga reverse engineering tool-chain: From bitstream to rtl code," *IEEE Access*, vol. 7, pp. 38379–38389, 2019.

[18] M. Ender, P. Swierczynski, S. Wallat, M. Wilhelm, P. M. Knopp, and C. Paar, "Insights into the mind of a trojan designer: The challenge to integrate a trojan into the bitstream," in *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, ASPDAC '19, (New York, NY, USA), pp. 112–119, ACM, 2019.

[19] A. Cui, C.-H. Chang, S. Tahar, and A. T. Abdel-Hamid, "A robust fsm watermarking scheme for ip protection of sequential circuit design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 5, p. 678–690, 2011.

[20] E. Castillo, A. H. C. García, L. Parrilla, D. P. Morales, A. Lloris, and U. Meyer-Bäse, "Digital signature embedding technique for ip core protection," *2007 3rd Southern Conference on Programmable Logic*, pp. 143–148, 2007.

[21] T. Nie, L. Zhou, and Y. Li, "Hierarchical watermarking method for fpga ip protection," 2013.

[22] Red Balloon Security Incorporated, "100 seconds of solitude." https://thrangrycat.com/, 2019.

[23] M. Schmid, D. Ziener, and J. Teich, "Netlist-level ip protection by watermarking for lut-based fpgas," pp. 209 – 216, 01 2009.

[24] California Northern District Court, "Konda technologies, inc. v. flex logix technologies, inc.." Case 5:18-cv-07581, 2019.

[25] Xilinx, "Vivado design suite user guide - partial reconfiguration." https://www.xilinx.com/support/documentation/sw_manuals/xilinx2019_1/ug904-vivado-implementation.pdf, 2017.

[26] P. Subramanyan, N. Tsiskaridze, K. Pasricha, D. Reisman, A. Susnea, and S. Malik, "Reverse engineering digital circuits using functional analysis," *Design, Automation Test in Europe Conference Exhibition*, 2013.

[27] J. J. Mcgregor, "Backtrack search algorithms and the maximal common subgraph problem," *Software: Practice and Experience*, vol. 12, no. 1, p. 23–34, 1982.

[28] D. Wang and L. Tian, "Parametric methods for confidence interval estimation of overlap coefficients," *Comput. Stat. Data Anal.*, vol. 106, pp. 12–26, Feb. 2017.

[29] Clifford Wolf, "PicoRV32 - A Size-Optimized RISC-V CPU." https://github.com/cliffordwolf/picorv32, 2019.

[30] OpenRISC Foundation, "OpenRISC1200 Implementation." https://github.com/openrisc/or1200, 2019.

[31] D. R. Foundation", "Darkriscv." https://github.com/darklife/darkriscv, 2019.

[32] OpenCores, "Sha cores." https://opencores.org/projects/shacore, 2018.