# New Security Threats on FPGAs: From FPGA Design Tools Perspective

Sandeep Sunkavilli, Zhiming Zhang, and Qiaoyan Yu
*Dept. of Electrical and Computer Engineering*
*University of New Hampshire*

***Abstract*—The growing market share of FPGAs motivates the increasing number of attackers to tamper with FPGA systems. The majority of existing research efforts on FPGA security focus on counterfeiting devices, hardware Trojans, reverse engineering hardware designs via decomposing or decrypting bitstream files, and side-channel analysis attacks. Those attacks are typically limited to the FPGA systems implemented in standalone FPGA devices. As more cloud-based FPGA providers, third-party accelerator suppliers, and open-source FPGA design tools are available for prototyping, hardware acceleration, and high-performance computing, new FPGA utilization models are gradually formed. The increasing number of entities involved in the new FPGA use model leads to the emergence of new security threats and attack surfaces. Although the security issues on FPGA systems design and piracy have been widely investigated, there is limited investigation available disclosing the security threats from the FPGA design tools perspective. This work conducts a comprehensive survey on the FPGA tool security and proposes a thorough security threat landscape for the new FPGA utilization model in the era of machine learning and cloud computing.**

***Index Terms*—FPGA security, hardware security, covert channel, hardware Trojan, multi-tenant FPGA, cloud computing.**

## I. Introduction

The growing market share of FPGAs motivates more and more attackers to tamper with either standalone or cloud-based FPGA systems. The majority of existing research efforts [1] on FPGA security focus on counterfeiting devices [2], hardware Trojans [3], [4], reverse engineering intellectual property (IP) designs via decomposing or decrypting bitstream files [5], and side-channel analysis attacks [6], [7]. Historically, the investigation on FPGA security is limited to the FPGA system implementations on standalone FPGA devices in the FPGA utilization model shown in Fig. 1(a). In that model, FPGA users have physical access to the FPGA device and its corresponding design suite. The regulation policies employed in the FPGA design flow are simple and easy to follow and track.

To facilitate computing-intensive applications such as machine learning, artificial intelligence, and cloud computing, FPGA design and utilization enter a new era, where more and more cloud-based FPGA providers [8] and third-party accelerators are integrated into the development flow of FPGA systems. A new FPGA utilization model shown in Fig. 1(b) gradually becomes appealing for prototyping, hardware acceleration, and high-performance computing. However, the increasing number of entities involved in the new FPGA utilization model leads to new potential attack surfaces. The

This work is partially supported by NSF award CNS-1652474.

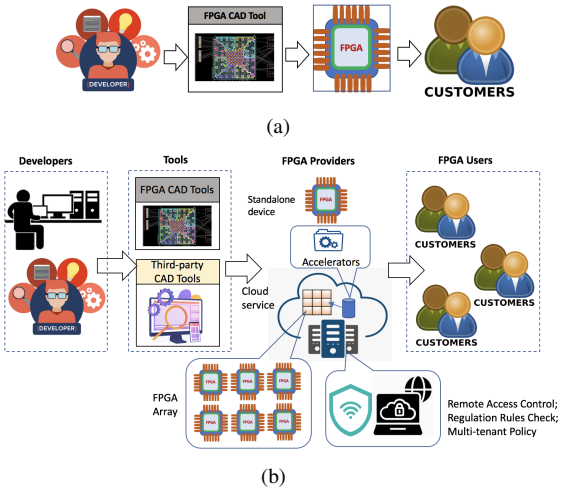



Fig. 1. FPGA utilization models. (a) Old model and (b) new model.

cloud FPGA providers enable FPGA users to perform computation remotely. As a result, some of the typical cybersecurity attacks will also apply to the FPGA system development flow. Sharing FPGA hardware in the cloud will further lead to design piracy from one FPGA customer to another [9], [10]. Recent literature demonstrates the feasibility of Rowhammer attack and power-based side-channel attack in the scenario of multi-tenant FPGAs [11], [12]. To increase the flexibility and compatibility, researchers start to explore open-source FPGA Computer-Aided Design (CAD) tools [13]–[15]. Since more developers for systems and third-party accelerators are involved in the new FPGA design flow, it is more challenging than before to ensure that all entities in the design flow are trustworthy. Untrusted CAD tools can perform malicious modifications to the FPGA configuration files [16], [17]. Moreover, the integration of third-party accelerators may need the traditional FPGA design suite to provide new interfaces to support verification and debugging. Attackers could leverage such interfaces to breach the integrity of the design flow.

The emerging new FPGA utilization model urges us to revisit the FPGA security. The recent surveys [18] provide a thorough overview of the attacks from the FPGA developers/users. There is limited study available revealing the potential risks induced by the tools for new FPGA system design. The main contributions of this work are as below:

- To the best of our knowledge, this is the first work that summarizes the potential security threats from commer-

cial and open-source FPGA CAD tools.

- A new security threat landscape is proposed for the emerging new FPGA utilization model.
- We analyze the new security vulnerabilities induced by the integration of countermeasures in FPGA CAD tools.

## II. Survey on Security Threats from FPGA Tools

### A. Security Threats from Commercial FPGA CAD Tools

To corrupt the original designs, malicious software can be embedded into the commonly used commercial CAD tools. The work [16] provides two attack examples, where the logic function of the original design is changed via the intermediate configuration files in the Xilinx ISE and Altera Quartus. Another work [19] indicates that the security of a complex FPGA-based System-on-Chip (SoC) can be corrupted by implanting malicious software to the FPGA CAD tools even if the protection of ARM TrustZone is employed in the system. The malicious CAD tools can attack the TrustZone by compromising the important communication signals, modifying the Verilog codes or FPGA Lookup Table (LUT) configurations, and changing the security status of IPs. By exploiting the padding schemes and analyzing the syntax errors reported by the target FPGA EDA tools, the work [20] demonstrates that the plaintext of the encrypted IPs can be recovered by attackers even if the IEEE P1735 standard is followed in the tools. Furthermore, hardware Trojans can be inserted to the IPs once the plaintext is extracted.

### B. Security Threats from Open-Source FPGA CAD Tools

The investigation of security threats and mitigation methods is typically tied with a specific FPGA chip and its design suite. It is not easy or feasible to migrate the risk assessment and defense methods to another system using other FPGAs from different vendors. A recent work [13] reveals the FPGA security threats from open-source FPGA CAD tools, VTR [14] and Symbiflow [15]. The findings from that work are valuable to promote the development of generalized methods for attack-resilient FPGA designs. The work [21] shows that a compromised design flow can insert a hardware Trojan, which is triggered by the action of bitstream generation in the design flow. This type of Trojans will not be detected by the traditional Trojan detection methods. As open-source FPGA CAD tools have gained increasing popularity due to their transparency and flexibility, it is imperative to perform more threat analyses and countermeasure development in an open-source CAD environment.

We envision that more attack surfaces could be developed in the open-source CAD tools than in the commercial FPGA design suites. Figure 2 summarizes the potential attack surfaces and the aims of possible attacks. First, the open-source tools often have a transparent flow for how the user constraints are employed in synthesis and floor planning. Some constraints defined by FPGA users could be tampered with or simply muted such that the pre-defined protection mechanisms are nullified. Second, more intermediate files in open-source CAD

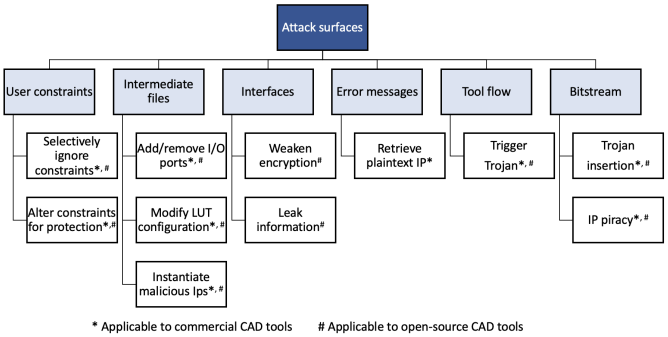



Fig. 2. Attack surfaces on commercial and open-source FPGA CAD tools.

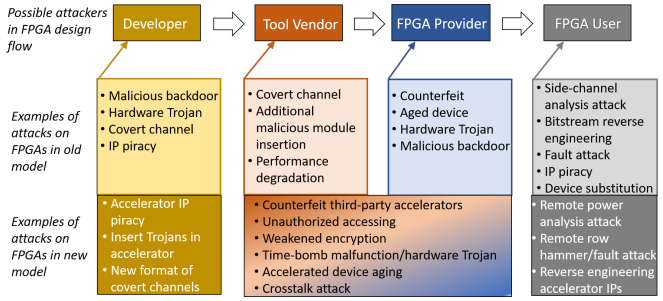



Fig. 3. Proposed new security threat landscape.

tools (e.g., *.blif*, *.net* and *.place* used in VTR and Symbiflow) are readable and editable before bitstream generation. The interfaces for third-party IP integration are more diverse than those in the commercial tools. FPGA system developers could leverage the interfaces to weaken the encryption on the licensed IPs or build covert channels for information leaking. Last but not least, the loosely protected toolchain will suffer from Trojan insertion and bitstream tampering [17], [22].

## III. Proposed New Security Threat Landscape

Due to the emergence of FPGA-as-a-service, more sectors are involved in the development flow of FPGA systems and thus the security threat landscape changes accordingly. Figure 3 summarizes the threat models induced by old and new FPGA utilization models. In the old FPGA utilization scenarios, *physical access* to the FPGA devices or CAD tools is required to execute the attack. Attacks are often localized either in the device and design suite vendor or in the end user. The most common attacks reported in the literature are (1) hardware Trojan insertion to build a covert channel or modify the original function at the developer end [4], [23], (2) counterfeit FPGA devices [2], and (3) side-channel analysis (SCA) attacks via power consumption or thermal observation at the FPGA user end [6], [7]. Few works disclose that some FPGA CAD tools can be tampered with to facilitate the implementation of covert channels and Trojan insertion [16], [17]. No matter which kind of attacks mentioned above is performed, collaborative attacks are less likely to happen because of the limited knowledge sharing among multiple sectors in the FPGA design flow.

As compared in Fig. 1, the new FPGA use model involves more entities than the old one. Thus, more security threats could be brought into the design flow. This trend will become severer as more and more open-source FPGA CAD tools are employed in the design flow. In cloud-based FPGA computing, the physical access to the CAD tools and FPGA devices are often prohibited; now, the new attack format could be a *synergy effort* deployed in the tools and devices. As a result, the attack model is shifted. As shown in Fig. 3, the attacks in the new FPGA utilization model from the tool vendors and FPGA providers may be combined. Open-source FPGA CAD tools are not customized for a particular FPGA chip, the architecture information and essential characteristics of the FPGA of interest should be available for the CAD tool. As discussed in Section II-B, the accessibility of intermediate files will further facilitate the occurrence of new attacks. For example, more counterfeit third-party accelerators [24] could be employed in the design flow. The diverse sources of IPs will challenge the authorized access control. The security strength provided by IP encryption will be weakened, as well. Information leaking via crosstalk and thermal covert channels [25], [26] could be more prevalent than those in the old FPGA use model. In addition, when we enter the era of FPGA-as-a-service, local SCA attacks are upgraded to remote versions, such as remote power analysis [12], remote rowhammer attack [11], and remote fault attack [27].

The new FPGA utilization model urges us to expand the threat model for new FPGA development flow. Our summary shown in Fig. 3 may not include all new attack examples reported in the existing literature. We hope that our work can inspire more engineers and researchers in the FPGA security community to enhance the knowledge and foresee the potential risks that we are facing in the new FPGA development scenarios. On the other hand, there are new opportunities for countermeasure design in open-source FPGA CAD tools. Various security features could be added to the conventional design flow by modifying and extending the vendor tools to secure FPGA systems against the new threats.

## IV. Integration of Defense Methods into FPGA Design Flow

### A. New Design Flow for FPGA Security

The existing defense methods against the security threats mentioned in the previous sections will influence the traditional FPGA design flow, either revising the existing synthesis and floorplanning algorithms or adding extra steps to perform encryption or obfuscation. Figure 4 highlights the design phases where different countermeasures can be integrated into the existing FPGA configuration flow. Two kinds of encryption, IP encryption and bitstream encryption, can be supported. The latter one is commonly used to thwart reverse engineering attacks. New constraints file scripts, synthesis and floorplanning algorithms are effective to isolate design modules, especially in multi-tenant FPGAs. To assure the isolation being activated in the design, the step *Design Rule Check* will reinforce the new policies for module isolation.

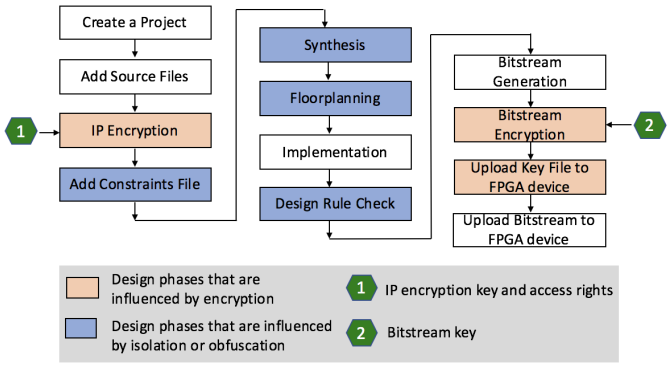



Fig. 4. New design flow after integrating countermeasures for FPGA security.

### B. Existing Integration of Countermeasures in FPGA Tools

*1) Bitstream Encryption:* Bitstream encryption is one of the popular countermeasures against hardware Trojan insertion and bitstream reverse engineering or tampering attacks. For instance, Xilinx 7-series FPGAs have an on-chip AES decryption module to authenticate the bitstream file. A 128-bit initial vector and a 256-bit key are provided to the Xilinx Vivado, which performs bitstream encryption after the phase of implementation [28]. Intel FPGAs are also equipped with an AES decryption module. Similarly, the Intel Quartus prime CAD tool uses a *Qcrypt* to encrypt bitstream [29].

*2) IP Encryption:* IP encryption allows IP owners to prohibit the unauthorized access to their IPs (i.e., determining how various vendor tools interact with their IPs), but it does not thwart any attacks performed after the bitstream file is generated. In the context of hardware implementation, IP encryption techniques are only applicable to the netlist described in the format of Verilog, SystemVerilog and VHDL. It is not suitable to perform IP encryption on the popular netlist format *EDIF*. The IEEE-1735 standard [30] has been incorporated in the commercial FPGA design suite, Xilinx Vivado, to support IP encryption. Three types of rights—common rights, vendor-specific rights and conditional rights—are defined along with the IP encryption option. Common rights apply to all the vendor tools specified by the IP owner; vendor specific rights are only for a specific vendor; conditional rights decide the level of access that will be given to the IP user. The IP user will be provided with an encrypted IP file, which consists of an encrypted public key and all the access rights defined by the IP owner.

*3) Design Isolation and Obfuscation:* The work [31] proposes to extend the routing algorithms in the FPGA CAD tools to protect FPGA designs from the crosstalk-based side-channel attacks. More specifically, the enhanced routing algorithm ensures that the nets carrying sensitive information, such as encryption keys, are never routed close to the untrusted nets, thus prohibiting the crosstalk attack. Another work [32] proposes a hardware isolation framework against information leaking (HILL) in multi-tenant FPGA systems. To mitigate the crosstalk attack, HILL reduces the usage of long wires for security-critical instances, isolates security-critical instances from other parts and tenants, and uses dummy long wires

to obfuscate the existing crosstalk side-channels. The authors of the work [32] claim that HILL can be integrated into any commodity FPGA CAD tools, such as Xilinx Vivado or Intel Quartus. The work [33] introduces an isolated partial reconfiguration design flow (IPRDF), which extends the Xilinx isolation design flow (IDF) [34] and is compatible with Xilinx vendor tools for bitstream generation. The partial reconfiguration provided by IPRDF can act as an effective countermeasure to single-event upsets (SEUs) and benefit the redundancy needed for security-critical systems. To prevent and tolerate the malicious tampering of critical functions, the work [35] uses an obfuscated voting unit (OVU) to provide the critical functions with multiple obfuscated variants, thus increasing the attack effort of the ergodic alterations from attackers. The work [36] proposes a logic obfuscation methodology for FPGA bitstreams, which inserts key bits to LUTs, to defend common bitstream attacks. That method is performed by a custom CAD framework and it can be integrated into the general FPGA design flow as an add-on step.

## V. Analysis of New Attack Surfaces Induced by Integrating Countermeasures in FPGA CAD Tools

Since encryption and isolation are two general categories of existing countermeasures, we analyze the potential risks of integrating those techniques into the FPGA CAD tools and demonstrate some case studies in this section.

### A. Security Vulnerability in IP Encryption

IP encryption restricts the access to the IP, depending upon the access rights defined by the IP owner. IP users only need the encrypted IP file, which has the information regarding the key and the access rights defined during encryption. IP users can only learn the access rights from the key file. If the encrypted IP will be used in a third-party tool, the public key of that tool should be provided before encryption. Both the keys are used in encryption. Any mismatch in the public key, the encrypted IP cannot be used. If the trustworthiness of the FPGA CAD tool is not guaranteed, there is a potential risk of key leaking.

IP encryption still suffers from the risk of IP piracy due to traceable pin names and schematic views. Even though an IP is encrypted, its net and pin names are still visible to the IP users. As shown in Fig. 5, the implementation of the encrypted IP is not readable, but the number of I/O pins, their width, and I/O names are observable. This fact leads to a potential risk. IP users could leverage this fact to trace the signals and precisely target the critical ones. When we zoom in the instantiated encrypted IP, the schematic view of the encrypted circuit will reveal the FPGA components used in the physical implementation of that IP module. For instance, Fig. 5 indicates that the hidden component used in the encrypted is a LUT. Moreover, it is possible to see the hierarchy of the encrypted module. Figure 6 shows a partially encrypted linear-feedback shift register (LFSR), where the encrypted submodules d1, d2, d3 and d4 are described in the format of encrypted netlist *dff.vp*. This example shows that we can

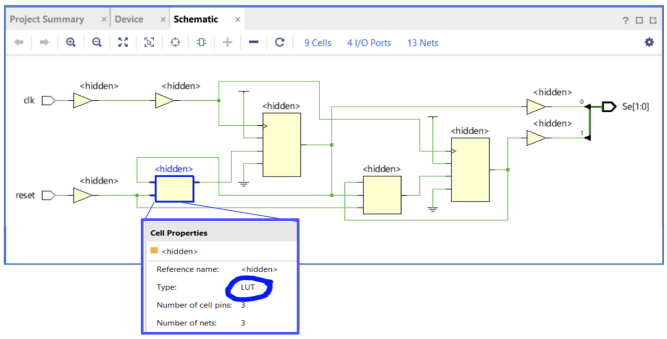



Fig. 5. Diagram of an instantiated encrypted counter.

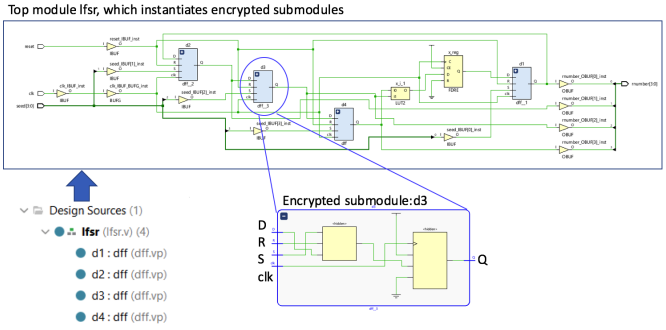



Fig. 6. Hierarchical view of a partially encrypted LFSR.

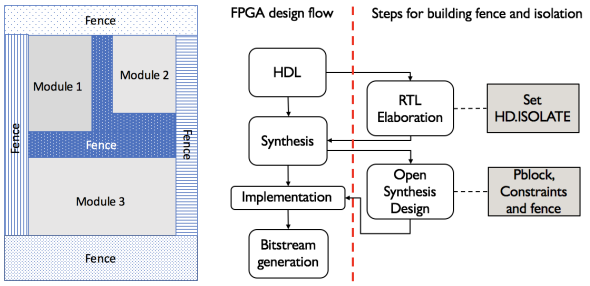
trace down the relation between the encrypted IPs and other submodules in the system design.

### B. Security Vulnerability in Design Isolation

Fence is a dedicated empty FPGA area that prohibits any logic from being implemented there. The purpose of fence is to isolate modules, thus thwarting the crosstalk attack. Figure 7(a) depicts an example of fence-based isolation. Xilinx introduces a concept called Isolation Design Flow (IDF) to physically and logically isolate modules with fence. The fence-based isolation requires a slight modification on the conventional FPGA design flow. As shown in Fig. 7(b), the two grey steps are added to the RTL elaboration and design synthesis phases. The property *HD.ISOLATE* directs the tool to create an isolated region *Pblock* for each module with this property. Xilinx *Vivado Isolation Verifier (VIV)* checks if the isolation fence is implemented successfully. Six Design Rule Check (DRC) rules [34] employed in Vivado can examine provenance and violations on I/O banks, package pins, floorplanning, placement and routing. However, the errors reported by DRC do not stop the designer from deploying the design into the FPGA device; instead, DRC only warns the designer to be aware of the violations.

The isolation fence only prevents hardware Trojan insertion in the fence area, but it cannot thwart malicious modification on the design due to the following limitations.

*1) Limitation 1: exception on global signals:* The security of fence-based isolation is originated from trusted routing, which ensures that each input is driven by one source and each output only drives one load. The IDF DRC will report




(a) (b)

Fig. 7. Isolation Design Flow (IDF) in FPGA design. (a) Fence implementation and (b) modified floorplanning in the design flow.

errors if there is any violation against trusted routing. However, there is an exception. The global signals in the modules requiring fence protection can be set with an isolation-exempt property *HD.ISOLATED EXEMPT* to pass the DRC without errors. This exempt property could be exploited by attackers to disguise the interconnect for a Trojan as global signals.

*2) Limitation 2: exception on non-isolated modules:* The fence-based isolation only prevents the illegal communication between isolated modules, not stopping the interaction between isolated and non-isolated modules. According to the IDF rules, a module without the HD.ISOLATE property will not be protected by a fence and thus it can be placed in any of the isolated regions (pblocks). In the design shown in Fig. 8(a), only M1, M2 and M3 with the isolation property will be configured in the three isolated FPGA regions, pblock1, pblock2 and pblock3, respectively. The remaining logic in the top module, including M4, can be placed in any of the three pblocks above. We performed a case study in Vivado to implement a covert channel, which aims for leaking key information from an AES encryption module. The baseline is a top module only carrying an AES unit. The isolation property is set to the AES unit to form an isolated region (pblock1). The output after implementation is shown in Fig. 8(b). Next, we introduced a hardware Trojan to the top module for the purpose of leaking the secret key from the AES unit. As the Trojan is the circuit under protection, we did not set the isolation property for the Trojan. Figure 8(c) indicates that the Trojan is successfully placed in the isolated region for AES. The center of the white interconnection is the Trojan. We further increased the number of pblocks in the top module. As shown in Fig. 8(d), we are able to insert the Trojan to the pblock where the AES unit is located. For both cases shown in Figs. 8(c) and (d), no IDF DRC error is reported to disclose the occurrence of the Trojan even though the entire IDF for the fence implementation is followed.

*3) Limitation 3: no protection mechanism at routing:* The instructions for fence implementation, isolation region creation, HD.ISOLATE property setting, and isolation region assignment for specific modules are stored in the constraints file. Figure 9 illustrates a snapshot for a constraints file, which is used in the process of synthesis and implementation. To place a Trojan in a design, modifying the constraints file is a vital step. Any personal/tool who has access to the constraints

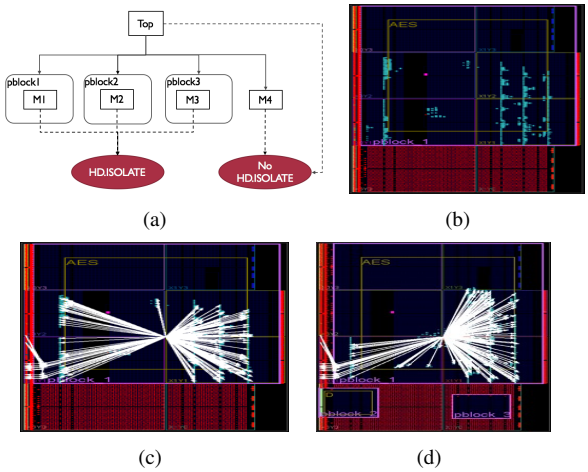



Fig. 8. An example for exception on non-isolated modules. (a) Top module overview of an IDF design, post-implementation device view for AES (b) without Trojans, with (c) a hardware Trojan in a single isolation pblock, and with (d) a hardware Trojan in one of the three pblocks.

```
set_property HD.ISOLATED true [get_cells AES]                    → HD.ISOLATED set for AES module
create_pblock pblock_1                                           → Isolated region pblock_1 created
add_cells_to_pblock [get_pblocks pblock_1] [get_cells -quiet [list AES]]
resize_pblock [get_pblocks pblock_1] -add {SLICE_X0Y199:SLICE_X89Y50}
resize_pblock [get_pblocks pblock_1] -add {BSCAN_X0Y0:BSCAN_X0Y3}
resize_pblock [get_pblocks pblock_1] -add {BUFGCTRL_X0Y0:BUFGCTRL_X0Y31}   } Range of the region
resize_pblock [get_pblocks pblock_1] -add {BUFHCE_X0Y12:BUFHCE_X1Y47}      (few lines from actual
resize_pblock [get_pblocks pblock_1] -add {BUFIO_X1Y4:BUFIO_X1Y11}         isolation block)
resize_pblock [get_pblocks pblock_1] -add {BUFMRCE_X0Y7:BUFMRCE_X1Y2}
.....
set_property PROHIBIT true [get_sites TIEOFF_X52Y178]
set_property PROHIBIT true [get_sites TIEOFF_X53Y178]
set_property PROHIBIT true [get_sites TIEOFF_X54Y178]
set_property PROHIBIT true [get_sites B9]                        } Fence (few lines from actual fence
set_property PROHIBIT true [get_sites C9]                          implementation)
set_property PROHIBIT true [get_sites OLOGIC_X0Y177]
set_property PROHIBIT true [get_sites ILOGIC_X0Y177]
set_property PROHIBIT true [get_sites OLOGIC_X0Y178]
.....
set_property PROHIBIT false [get_sites B9]                       → Pin freed from Fence by changing prohibit value
```

Fig. 9. A snapshot of a constraints file in IDF.

file can make changes on the constraint setting, as there is no integrity check available for the constraints file. To assure the success of the Trojan insertion in the case study mentioned in Section V-B2, we modified the pin prohibit values from *true* to *false* by altering constraints shown in the last line in Fig. 9.

*4) Limitation 4: unutilized resources:* The FPGA resources allocated for an isolation region are not only usable for the module to be isolated, but also accessible for the remaining logic without isolation property in the top module. As the prediction on the size of the isolation region for the module under protection is not always precise, there are unutilized FPGA resources available for attackers to implement hardware Trojans. We resume our case study mentioned in Section V-B2. Table I reports the FPGA resources allocated for the AES pblock without/with the Trojan. As shown, the hardware Trojan only increases the LUT utilization, LUTRAM, and flip flop (FF) by 0.02%, 0.03%, 0.03%, respectively. No changes are observed in the utilization of BRAM and BUFG. Due to the communication, the Trojan indeed increases the number of I/O. From this comparison, we also conclude that the fence implementation is very likely to result in significant FPGA resource wasting.

TABLE I
RESOURCE UTILIZATION IN OUR CASE STUDY OF TROJAN INSERTION.

| Resource | Available | Utilization without Trojan | Resource utilization rate without Trojan | Utilization with Trojan | Resource utilization rate with Trojan |
|---|---|---|---|---|---|
| LUT | 47000 | 1128 | 2.40% | 1137 | 2.42% |
| LUTRAM | 14400 | 0 | 0% | 4 | 0.03% |
| FF | 94000 | 1169 | 1.24% | 1193 | 1.27% |
| BRAM | 105 | 68 | 64.76% | 68 | 64.76% |
| I/O | 2/14 | 2 | 100% | 14 | 100% |
| BUFG | 32 | 1 | 3.13% | 1 | 3.13% |

## VI. CONCLUSION

The prevalent cloud-based FPGA services and open-source FPGA CAD tools gradually change the traditional FPGA design and use model. More and more entities are involved in the development flow of FPGA systems. The newly emerged FPGA use model poses new and unique challenges to FPGA security. This work complements the existing surveys on the attacks from FPGA developers and users by investigating the potential security threats from the commercial and open-source FPGA CAD tools. A comprehensive landscape for the new security threats is proposed in this work. Furthermore, this work analyzes the new security vulnerabilities induced by the integration of defense methods into the typical FPGA design flow. Several case studies are provided accordingly to inspire researchers to reconsider the new security issues that may occur in the deployment of countermeasures into FPGA CAD tools, especially when we implement more FPGA applications in the new FPGA utilization model.

## REFERENCES


[1] S. M. Trimberger and J. J. Moore, "FPGA Security: Motivations, Features, and Applications," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1248–1265, 2014.

[2] H. Dogan, D. Forte, and M. M. Tehranipoor, "Aging Analysis for Recycled FPGA Detection," in *Proc. DFT'14*, pp. 171–176, 2014.

[3] J. Zhang and G. Qu, "Recent Attacks and Defenses on FPGA-Based Systems," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 12, Aug. 2019.

[4] S. Gundabolu and X. Wang, "On-chip Data Security Against Untrustworthy Software and Hardware IPs in Embedded Systems," in *Proc. ISVLSI'18*, pp. 644–649, 2018.

[5] 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.

[6] L. Wei, B. Luo, Y. Li, Y. Liu, and Q. Xu, "I Know What You See: Power Side-Channel Attack on Convolutional Neural Network Accelerators," in *Proc. ACSAC'18*, pp. 393–406, 2018.

[7] M. Thoonen, "Hardening FPGA-based AES implementations against side channel attacks based on power analysis," B.S. thesis, University of Twente, 2019.

[8] S. Narula, A. Jain, and Prachi, "Cloud Computing Security: Amazon Web Service," in *Proc. ICACCT'15*, pp. 501–505, 2015.

[9] J. Krautter, D. R. E. Gnad, and M. B. Tahoori, "Mitigating Electrical-Level Attacks towards Secure Multi-Tenant FPGAs in the Cloud," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 12, Aug. 2019.

[10] I. Giechaskiel, K. Eguro, and K. B. Rasmussen, "Leakier Wires: Exploiting FPGA Long Wires for Covert- and Side-Channel Attacks," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 12, Aug. 2019.

[11] J. Krautter, D. R. Gnad, and M. B. Tahoori, "FPGAhammer: Remote Voltage Fault Attacks on Shared FPGAs, suitable for DFA on AES," *TCHES*, pp. 44–68, 2018.

[12] F. Schellenberg, D. R. Gnad, A. Moradi, and M. B. Tahoori, "An Inside Job: Remote Power Analysis Attacks on FPGAs," in *Proc. DATE'18*, pp. 1111–1116, 2018.

[13] S. Sunkavilli, Z. Zhang, and Q. Yu, "Analysis of Attack Surfaces and Practical Attack Examples in Open Source FPGA CAD Tools," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, pp. 504–509, 2021.

[14] K. E. Murray, O. Petelin, S. Zhong, J. M. Wang, M. Eldafrawy, J.-P. Legault, E. Sha, A. G. Graham, J. Wu, M. J. P. Walker, H. Zeng, P. Patros, J. Luu, K. B. Kent, and V. Betz, "VTR 8: High-Performance CAD and Customizable FPGA Architecture Modelling," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 13, May 2020.

[15] K. E. Murray, M. A. Elgammal, V. Betz, T. Ansell, K. Rothman, and A. Comodi, "SymbiFlow and VPR: An Open-Source Design Flow for Commercial and Novel FPGAs," *IEEE Micro*, vol. 40, p. 49–57, July 2020.

[16] Z. Zhang, L. Njilla, C. A. Kamhoua, and Q. Yu, "Thwarting Security Threats from Malicious FPGA Tools with Novel FPGA-Oriented Moving Target Defense," *TVLSI*, vol. 27, no. 3, pp. 665–678, 2019.

[17] R. S. Chakraborty, I. Saha, A. Palchaudhuri, and G. K. Naik, "Hardware Trojan Insertion by Direct Modification of FPGA Configuration Bitstream," *IEEE Design Test*, vol. 30, no. 2, pp. 45–54, 2013.

[18] C. Jin, V. Gohil, R. Karri, and J. Rajendran, "Security of Cloud FPGAs: A Survey," *arXiv preprint arXiv:2005.04867*, 2020.

[19] E. M. Benhani, L. Bossuet, and A. Aubert, "The Security of ARM TrustZone in a FPGA-Based SoC," *IEEE Transactions on Computers*, vol. 68, no. 8, pp. 1238–1248, 2019.

[20] A. Chhotaray, A. Nahiyan, T. Shrimpton, D. Forte, and M. Tehranipoor, "Standardizing Bad Cryptographic Practice: A Teardown of the IEEE Standard for Protecting Electronic-Design Intellectual Property," in *Proc. CCS'17*, p. 1533–1546, 2017.

[21] C. Krieg, C. Wolf, and A. Jantsch, "Malicious LUT: A Stealthy FPGA Trojan Injected and Triggered by the Design Flow," in *Proc. ICCAD'16*, 2016.

[22] A. Moradi and T. Schneider, "Improved Side-Channel Analysis Attacks on Xilinx Bitstream Encryption of 5, 6, and 7 Series," in *Proc. COSADE'16*, pp. 71–87, 2016.

[23] S. Mal-Sarkar, A. Krishna, A. Ghosh, and S. Bhunia, "Hardware Trojan Attacks in FPGA Devices: Threat Analysis and Effective Countermeasures," in *Proc. GLSVLSI'14*, pp. 287–292, 2014.

[24] F. Turan and I. Verbauwhede, "Trust in FPGA-accelerated Cloud Computing," *CSUR*, vol. 53, no. 6, pp. 1–28, 2020.

[25] G. Provelengios, C. Ramesh, S. B. Patil, K. Eguro, R. Tessier, and D. Holcomb, "Characterization of Long Wire Data Leakage in Deep Submicron FPGAs," in *Proc. FPGA'19*, pp. 292–297, 2019.

[26] S. Tian and J. Szefer, "Temporal Thermal Covert Channels in Cloud FPGAs," in *Proc. FPGA'19*, p. 298–303, 2019.

[27] M. M. Alam, S. Tajik, F. Ganji, M. Tehranipoor, and D. Forte, "RAM-Jam: Remote Temperature and Voltage Fault Attack on FPGAs using Memory Collisions," in *Proc. FDTC'19*, pp. 48–55, 2019.

[28] Xilinx, "Using Encryption to Secure a 7 Series FPGA Bitstream," *https://www.xilinx.com/support/documentation/application_notes/xapp1239-fpga-bitstream-encryption.pdf*, vol. v1.2, 2021.

[29] Intel, "AN 556: Using the Design Security Features in Intel FPGAs," *https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/an/an556.pdf*, vol. 11.12, 2019.

[30] "IEEE Recommended Practice for Encryption and Management of Electronic Design Intellectual Property (IP)," *IEEE Std 1735-2014 (Incorporates IEEE Std 1735-2014/Cor 1-2015)*, pp. 1–90, 2015.

[31] Z. Seifoori, S. S. Mirzargar, and M. Stojilović, "Closing Leaks: Routing Against Crosstalk Side-Channel Attacks," in *Proc. FPGA'20*, p. 197–203, 2020.

[32] Y. Luo and X. Xu, "HILL: A Hardware Isolation Framework Against Information Leakage on Multi-Tenant FPGA Long-Wires," in *Proc. ICFPT'19*, pp. 331–334, 2019.

[33] K. Pham, E. Horta, D. Koch, A. Vaishnav, and T. Kuhn, "IPRDF: An Isolated Partial Reconfiguration Design Flow for Xilinx FPGAs," in *Proc. MCSoC'18*, pp. 36–43, 2018.

[34] S. Pitaka, "Isolation Design Flow for Xilinx 7 Series FPGAs or Zynq-7000 SoCs (Vivado Tools)," *Xilinx XAPP1222*, 2020.

[35] T. Hoque, K. Yang, R. Karam, S. Tajik, D. Forte, M. Tehranipoor, and S. Bhunia, "Hidden in Plaintext: An Obfuscation-Based Countermeasure against FPGA Bitstream Tampering Attacks," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 25, Nov. 2019.

[36] B. Olney and R. Karam, "Tunable FPGA Bitstream Obfuscation with Boolean Satisfiability Attack Countermeasure," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 25, Feb. 2020.