# Poster: Towards Reverse Engineering FPGA Bitstreams for Hardware Trojan Detection

Yezee Seo[1], Junghwan Yoon[1], Jaedong Jang[1], Mingi Cho[1], Hoon-Kyu Kim[2], and Taekyoung Kwon[1]

[1]Information Security Lab, Yonsei University, Seoul, 03722, Korea
[2]Agency for Defense Development, Seoul, Korea
[1]{seoyz0716, yjh1226, woehd91, imgc, taekyoung}@yonsei.ac.kr
[2]hunk@add.re.kr

***Abstract*—FPGAs are field-programmable and reconfigurable integrated circuits, aiming at both hardware and software advantages. They recently tend to combine with microprocessors in the form of all programmable SoCs. A security problem in FPGAs is that the configuration data called a bitstream, which must be loaded to circuits, is susceptible to both malicious fabrication and modification attacks due to flexibility. That is, a hardware Trojan can be loaded to the circuits. In this study, we consider a reverse engineering of bitstreams promising for hardware Trojan detection in a static manner because modern techniques relying on dynamic signal analysis are not cost-effective nor precise. A challenge is that the reverse engineering of bitstreams is not relatively easy and that the detailed format of the bitstream is proprietary to the FPGA vendors. As a preliminary study, we design the general architecture of bitstream reverse engineering for hardware Trojan detection in this respect, and present a detailed method for reverse engineering the core resources of FPGAs. We also discuss our on-going work and future directions.**

## I. Introduction

A field-programmable gate array (FPGA) is an integrated circuit device that can be programmed and also be re-programmed after manufacture to run many specific applications. It can also implement software processor cores and combine with hardware processor cores. These reconfigurable and general features of FPGAs allow designing an application system more flexible, expecting both hardware performance and software diversity in FPGAs. For the reasons, FPGAs are already used in various application fields, such as cryptographic core, multimedia processing, automotive, and military systems, and the fields employing FPGAs are still growing. The system loaded onto the FPGA is first programmed in hardware description language (HDL), such as Verilog and VHDL, and then the synthesized design is loaded onto the FPGA device in the form of a bitstream. During the synthesis process, various external IP cores might be employed mostly in the way of protecting those IPs of the third party.

As the use of FPGAs magnificently increases, there are many growing concerns about security of FPGAs because of potential threats, such as hardware Trojan (HT), cloning, tampering and denial of service attacks [8]. HT is a real malicious threat because it can hide in hardware avoiding trivial dynamic detection methods until launched, and if conditioned, perform many kinds of malicious actions, such as information leakage and unintentional malfunctions [4].

In FPGA-based systems, the HT could be inserted into the FPGA design through many routes, e.g., outsourcing to external vendor, using untrusted third-party IPs, and reconfiguring in the FPGA supply chain. To cope with these problems, various methods for HT detection have been studied and also applied. Interestingly, most of those approaches rely on logic testing and side-channel analysis. Saying, they are dynamic analysis methods to detect HT by observing the signals obtained by specific device when HT is activated. Thus, there remain limitations: logic testing is difficult to trigger HT, and side-channel analysis is not easy to detect HT if the sensible effect of HT is insignificant [2]. To overcome such limitations, a detection method based on static analysis was also studied but with a gate-level netlist given [6]. Such static analysis methods have difficulty in detecting HT inserted directly into the bitstreams through modification or manipulation of the existing bitstreams. Therefore, to detect HT, it is necessary to "reverse engineer" the bitstream to the gate-level netlist. However, it is a challenging task to perform bitstream reverse engineering (RE) because vendors are reluctant to disclose the bitstream format and the design size and complexity of FPGAs have significantly increased.

The previous studies of FPGA bitstream RE aim at bitstream format analysis and efficient reconfiguration. *debit* [5] first introduced a correlation algorithm for bitstream RE by analyzing the bitstream format of Virtex 2. *BIL* [1] extended the previous work by employing as pre-knowledge the XDLRC file which contains information about all resources in order to evaluate the result of RE. Although *BIL* recovered partial resources of specific tiles only, it showed a promising direction for bitstream RE. *bit2ncd* [3] aimed at more complete RE for efficient reconfiguration purposes although it was unclear whether the logic implemented in each lookup table (LUT) was correctly recovered. Unlike the previous RE studies, in this paper, we are focused on hardware Trojan detection.

## II. System Design

Given a bitstream file, we need to recover a netlist that reveals the actual FPGA circuit configuration. The configuration resources are clearly divided into two parts: Programmable Interconnect Point (PIP) that represents the connection information of FPGAs, and Programmable Logic Point (PLP) that shows logic implementation such as clocks, multipliers, registers, and LUTs. Thus, we need to recover them for HT detection. Among various form of netlist, we consider a textual format, such as XDL, to identify configurable functions and finally use them for static analysis and HT detection. We adopt a machine learning technique for HT detection in XDL level,

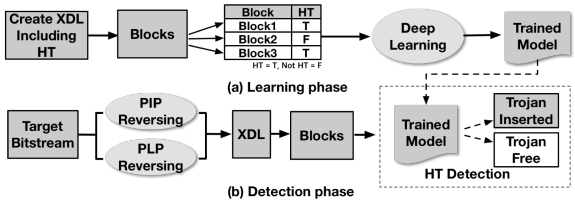


Fig. 1: System Architecture

such as deep neural networks to learn HT configurations and to perform classification. Fig. 1 illustrates the system architecture and the overall concept.

### A. PIP Reverse Engineering

We adopt *BIL*, which is the most promising RE work on PIP, in our study. *BIL* implemented an algorithm for finding bit offsets to recover PIPs in INT tiles. To be useful for HT detection, there are three challenges in this method. First, *BIL* uses only a single pair of bit and XDL files for DB construction. Thus, it can partially cover PIPs used in the source XDL file. Second, *BIL* deals with a single PIP, and so it cannot cover PIPs that are paired. Third, there remain PIPs represented as 'Zeroed' which means *BIL* could not deal with. Thus, we begin with improving *BIL* for PIP RE. We make it possible to employ multiple pairs of bit and XDL files in DB construction, to find more PIPs on RE. For zeroed PIPs, we create a mapping table with the information that the start wire or the end wire of zeroed PIPs are connected to different PIPs in the tile. We also use multiple pairs of bit and XDL files to validate the mapping table. In this stage, DB is constructed by removing the errors caused by PIPs in the mapping table.

### B. PLP Reverse Engineering

Logic circuits designed by a developer are implemented in LUT which is one of the PLP resources. Thus, to understand the logic implemented in FPGAs, it is important to reverse engineer the logic information assigned in the LUT. In order to reverse engineer the LUT, we need to find the corresponding offsets. We adopt the generic method introduced in [7] to find all offsets corresponding to the LUT data. Unlike [7], we only compare the configuration data field of each bitstream. When a bitstream is loaded into a FPGA, configuration data of bitstream is divide into frames which have their own addresses, and each frame configures specific configurable points. Therefore, by extracting and comparing the configuration data, it is possible to know which frames configure the LUT. So we extract the configuration data from each bitstream and compare them all. We use the data of LUT storing a value set to 1 in only one bit of 64-bit to recover the originally stored value from extracted data. By `OR`ing the data which the bit of the same index with extracted data is set, we can recover the originally stored value. After extracting data and recovering the stored value in the LUT, we convert it into a boolean equation by reconstructing the truth table.

## III. Implementation and Evaluation

We implement the basic system and evaluate it on the `xc5vfx30t-ff665` model of the Virtex-5 family. For PIPs, we evaluate the RE result using 18 samples. The database construction rate of the INT tile has increased from 87% to 93.6%. In addition, we newly found 80 cases that two PIPs always paired in use and 7 cases that three PIPs always used together. We recovered, albeit partially, the zeroed PIPs using our mapping table. As a result, the average recovery rate of INT tiles has increased from 82% to 89.4%. We may increase those rates by employing more pairs for database construction.

For LUTs, we basically utilized the *bitextract* module and wrote a python code to compare the configuration data. From the comparison result, we found that the LUT data in slice with an odd $x$ coordinate is located in frames #26 to #29, and the LUT data in slice with an even $x$ coordinate is located in frames #32 to #35 among the 36 frames of a CLB block. Based on this result, we obtained all data offsets of 20,480 LUTs in the device used in our experiment. To verify the extracted LUT data, we stored a random value in each LUT using verilog and extracted the data corresponding to the LUT. As a result of using `OR` operation, we were able to confirm that the originally stored value was correctly recovered. Finally we reconstructed the truth table with recovered values, and converted it into boolean equations. Each converted equation has the same functionality with original equation.

## IV. Summary and Future Direction

As a preliminary study of HT detection in the bitstream level of FPGAs, we addressed how to reverse engineer bitstreams for HT detection, and particularly we mainly showed how to recover the PIP and LUT information from bitstreams. Through the improvement, we increased the recovery rate of PIP and succeeded in extracting the boolean equation from the LUT data. The further step of our study is detecting the HT through the recovered XDL file. For HT detection from recovered netlists, we need to construct a database which consists of the static features extracted from the collected HT dataset in the same netlist level. We plan to apply a static analysis and a machine learning based detection technique that is popular in a software community for malware detection.

## Acknowledgment

This work was supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract (UD160066BD).

## References

[1] F. Benz, A. Seffrin, and S. A. Huss, "BIL: A Tool-chain for Bitstream Reverse-engineering," in *Proc. The International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE, 2012, pp. 735–738.

[2] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware Trojan Attacks: Threat Analysis and Countermeasures," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1229–1247, 2014.

[3] 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.

[4] V. Jyothi and J. J. Rajendran, "Hardware Trojan Attacks in FPGA and Protection Approaches," in *The Hardware Trojan War*. Springer, 2018, pp. 345–368.

[5] J.-B. Note and É. Rannaud, "From the Bitstream to the Netlist." in *Proc. ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, vol. 8, 2008, pp. 264–264.

[6] H. Shen and Y. Zhao, "HTChecker: Detecting Hardware Trojans Based on Static Characteristics," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2017, pp. 1–4.

[7] P. Swierczynski, M. Fyrbiak, P. Koppe, and C. Paar, "FPGA Trojans Through Detecting and Weakening of Cryptographic Primitives," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 8, pp. 1236–1249, 2015.

[8] 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.

# Poster: Towards Reverse Engineering FPGA Bitstreams for Hardware Trojan Detection

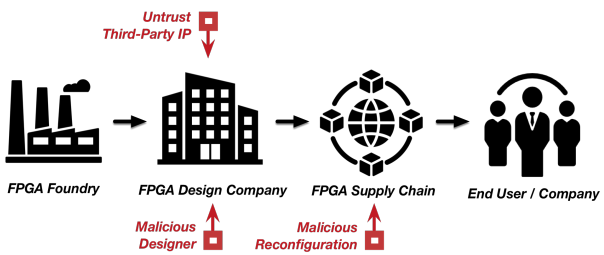
Yezee Seo[1], Junghwan Yoon[1], Jaedong Jang[1], Mingi Cho[1], Hoon-Kyu Kim[2], and Taekyoung Kwon[1]

1 Information Security Lab, Yonsei University, Seoul, 03722, Korea

2 Agency for Defense Development, Seoul, Korea

## Motivation

- Due to the flexibility of programming and the possibility of reconfiguration, FPGAs are widely used in many fields.
- The hardware trojan (HT) could be inserted into the FPGA design through many routes, e.g., outsourcing to external vendor, using untrusted third-party IPs, and reconfiguring in the FPGA supply chain.




- Most approaches rely on **logic testing** and **side-channel analysis**. There remain **limitations**: logic testing is difficult to trigger HT, and side-channel analysis is not easy to detect HT if the sensible effect of HT is insignificant.
- The previous static analysis methods have difficulty in detecting HT inserted directly into the bitstream through modification or manipulation of the existing bitstream.

**Therefore, to detect HT, it is necessary to "reverse engineer" the bitstream to the gate-level netlist.**

- **Related work**
  - `Note et al (FPGA 2008)` performs bitstream reverse engineering (RE) by analyzing the bitstream format.
  - `F. Benz et al (FPL 2012)` performs bitstream RE to confirm whether complete bitstream RE is feasible.
  - `Ding et al (MICPRO 2013)` performs bitstream RE for efficient reconfiguration purposes.

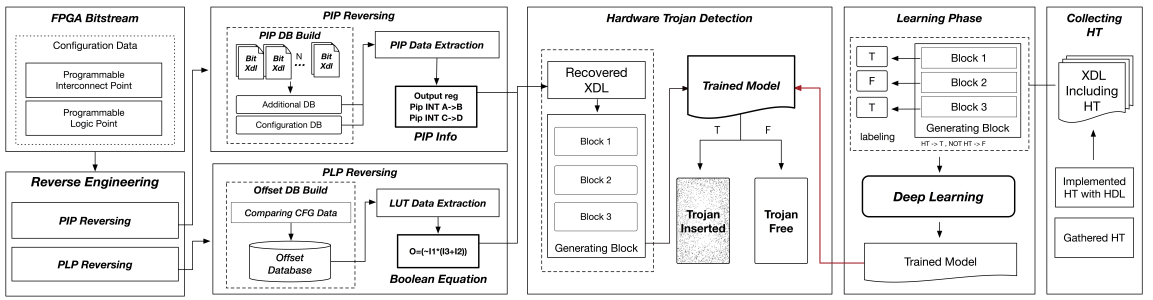
**The previous studies of FPGA bitstream RE aim at bitstream format analysis and efficient reconfiguration.**

## Our Objective

- We aim to **reverse engineer** the programmable **interconnect point (PIP)** and **programmable logic point (PLP)** information from the bitstream to gate-level netlist for HT detection.

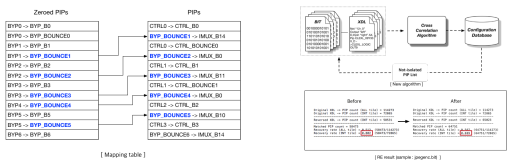
Acknowledgement: This work was supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract (UD160066BD).

## System Design

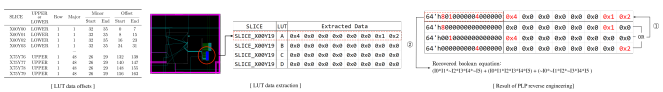



## Implementation & Evaluation

- **Target FPGA device** : Xilinx Virtex-5 xc5vfx30t-ff665
  - **PIP :** The DB construction rate of the INT tiles has reached to 93.6% using 314 pairs of bit and XDL files.
  - We recovered, albeit partially, the zeroed PIPs using our mapping table. As a result, the average recovery rate of INT tiles has increased to 89.4%. (less than 30 sec for INT tile PIPs, 1,651kB).

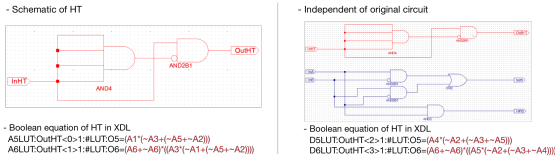



  - **PLP - Lookup Table (LUT) :** We utilized the *BIL* and python to compare the configuration data. As a result, we could obtain all the offsets of LUTs in the device.
  - We converted the extracted LUT data into a boolean equation and compared it with the original one. Note that the functionalities of the two equations are the same.

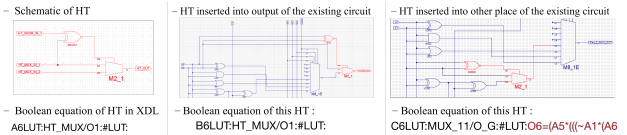



## HT Types Modifying the Bitstream

- **Type-1 Trojans**
  - Def: HT bitstream parts are inserted into the existing bitstream at empty positions (where no resources are utilized).
  - Independent of the original circuit (**no interconnection**)
  - RE result: The boolean equations of HT are preserved.




- **Type-2 Trojans**
  - Def: HT bitstream parts are inserted into the existing bitstream and so **interconnected** to the original circuitry.
  - RE result: The boolean equations of HT are preserved only when the HT was inserted into the output of the existing circuitry. However, they are **not** preserved in other cases.




## Summary & Future Direction

- **Summary**
  - We designed the basic system architecture for static HT detection at the bitstream level of FPGAs.
  - We addressed the significance of bitstream reverse engineering for HT detection as static analysis.
  - We showed the preliminary experimental study of reversing PIPs and LUTs from bitstreams.
- **Future Direction**
  - The further step of our study is detecting the HT through the recovered XDL file. For HT detection from recovered netlists, we need to construct a database which consists of the static features extracted from the collected HT dataset in the same netlist level.
  - We plan to apply a static analysis and a machine learning based detection technique that is popular in a software community for malware detection.

* Corresponding author's email: taekyoung@yonsei.ac.kr