NAECON 2023 - IEEE National Aerospace and Electronics Conference | 979-8-3503-3878-2/23/$

Determining Confidence in FPGA Bitstream Reverse Engineering Results

Ronald D. Williams
Dept. of ECE
University of Virginia
Charlottesville, VA, USA
rdw@virginia.edu

Zachary A. Collier
Dept. of Management
Radford University
Radford, VA, USA
zcollier@radford.edu

Thomas L. Polmateer
Dept. of Systems Engineering
University of Virginia
Charlottesville, VA, USA
polmateer@virginia.edu

James H. Lambert
Dept. of Systems Engineering
University of Virginia
Charlottesville, VA, USA
lambert@virginia.edu

Anvesh Perumalla
Dept. of ECE
University of Cincinnati
Cincinnati, OH, USA
perumaak@mail.uc.edu

John M. Emmert
Dept. of ECE
University of Cincinnati
Cincinnati, OH, USA
john.emmert@uc.edu

***Abstract*—*Reverse engineering (RE) is a widespread practice within engineering, and it is particularly relevant for discovering malicious functionality in digital hardware components. In this paper, we discuss bitstream or firmware RE for field programable gate arrays (FPGAs). A bitstream establishes the configuration of the FPGA device fabric. Complete knowledge of both the physical device fabric and a specific bitstream should be sufficient to determine the complete configuration of the programmed FPGA. However, a significant challenge to bitstream RE arises because information about the FPGA fabric and interpretation of the bitstream is typically incomplete. The uncertainties limit the confidence in the correctness of any configuration determined through the RE process. This paper identifies representative sources of uncertainty in bitstream RE of FPGA devices.*

Keywords*— *Bitstream; Confidence; FPGA; Reverse Engineering; Uncertainty

I. Introduction

The practice of reverse engineering (RE) often carries a negative connotation associated with the theft of intellectual property (IP). For example, concerns have been recently raised about RE of U.S. military equipment left after the withdrawal from Afghanistan by the nation's adversaries [1]. However, while RE is sometimes associated with IP infringement, security risks, and counterfeiting, it can also be used for legitimate applications, aiding in detection of counterfeits and malicious insertions, and defending IP claims [2-3].

Several general definitions of RE have been proposed [4-6]. A standard legal definition of RE is "starting with the known product and working backward to divine the process which aided in its development or manufacture" [6]. Reverse engineering can also be considered a "special case of systems engineering" in which one attempts to define the functional and dimensional specifications across the structural hierarchy of the item and its elements [4]. Reverse engineering seeks to discover and document these specifications by performing tests, measurements, and analyses which may be destructive or nondestructive in nature [4]. The Semiconductor Chip Protection Act of 1984 recognizes certain legitimate purposes of RE such as "teaching, analyzing, or evaluating the concepts or techniques embodied" in the technology [6]. Trust in digital hardware is central to everything that involves computing, and relative to integrated circuits (ICs), RE is an important area of security research.

Field programmable gate array (FPGA) digital hardware is widely used because it offers the flexibility of simple configuration. However, the advantage of easy configuration can become a liability exposing a system to attack at many places along the supply chain and in the deployed system. Reverse engineering of the device configuration file offers the possibility of extracting the behavior and structure of the device to expose the device functionality including any malicious functionality. Reverse engineering of the device configuration file can provide confirmation of expected functionality, understanding of unknown functionality, and exposure of undesired functionality [7]. An example of unintended functionality is the insertion of malicious alterations known as hardware Trojans. These Trojans have a number of potentially negative effects, including alteration or denial of service of IC functionality, reduction of IC reliability, and exfiltration of sensitive information [8]. Due to the increased outsourcing of phases of the IC production process (e.g., design, synthesis, fabrication, distribution), there are ample opportunities for the insertion of hardware Trojans [8-9]. FPGAs are particularly a target due to their reconfigurability once deployed in the field, and they are especially vulnerable during firmware upgrades [10].

In the remainder of the paper, we discuss a specific element of RE, i.e., that of bitstream RE of field configurable digital hardware, and the associated challenges of establishing confidence in the results of such RE activities.

This effort was supported by the National Science Foundation under Grants 1916760 and 1916722, Phase 1 IUCRC: Center for Hardware and Embedded Systems Security and Trust (CHEST)".

A. Bitstream Reverse Engineering

A bitstream establishes the configuration of an SRAM-based field configurable device fabric into a specific functional logic element. The device fabric is the static physical structure of the device including all logic components and interconnect. The bitstream contains the information necessary to configure the device fabric to perform logic operations. Complete knowledge of both the physical device fabric and a specific bitstream should be sufficient to determine the complete configuration of the SRAM-based field programmable device thus defined by the bitstream.

Software RE is a more studied topic than bitstream or firmware RE [11]. The conversion of a bitstream into a netlist is more complicated because of limited available information mapping bitstream segments into specific netlist configurations [12]. The additional challenges to bitstream RE introduce uncertainties into the extracted netlist [13], and these uncertainties reduce the confidence in the correctness of the results. While various tools are being developed to support bitstream RE, these tools typically exhibit low accuracy while demanding much computational time [14]. In addition, existing tools target one device family at a time, and are not easily modified to support other FPGA manufacturers and families [15-24].

This paper identifies representative sources of uncertainty in bitstream RE for generic and special FPGA architectures [26]. A more comprehensive description of the FPGA hardware is found in [26]. This identification of uncertainty sources can be used to inform the collection of evidence to support confidence in the result.

II. Configuration Overview

For most FPGAs, configuration is performed by loading data into a frame-based programming memory. This data is contained in a bitstream. Typically, the device will include a serial path to minimize the number of pins used for bitstream loading. Alternately, parallel paths may be available to provide higher performance or greater interface flexibility. For example, the Xilinx® 7 series devices provide a serial interface and parallel interfaces having widths of 8, 16, or 32 bits. The example devices have fixed length bitstreams for each device type [27-28].

A. Generic Bitstream

The bitstream contains both the information for configuring the fabric and the information on how to perform the configuration. The bitstream format is specific to the architecture, and this section provides information about the type of information that would typically be found in a bitstream. A bitstream is conveyed using a protocol defined by the device manufacturer. A significant part of the bitstream RE process involves the exposure of this protocol.

B. Configuration Packets

The parts of the bitstream that are logically interpreted are configuration packets. Project X-Ray [29] found three types of packets for the Xilinx bitstreams. Type 0 packets zero-fill between rows. Type 1 packets are used for reads and writes. Type 2 packets extend the number of words for type 1 packets. All packets include a header that includes one of three commands: NOP, READ, and WRITE. The header also includes other information such as an address and a word count. Bitstreams for different devices from different manufacturers will follow different protocols.

C. Configuration Registers

The address field in the configuration packet selects a particular configuration register. Data written to a configuration register provides control over the device and the configuration sequence.

D. Absence of Details

The documentation provided by the manufacturer provides a high-level view of the device configuration process. This documentation allows examination of the bitstream to determine the general structure of the bitstream. Unfortunately, the information provided is typically insufficient to determine the device configuration defined by the bitstream.

III. Confidence in Reverse Engineering

The objective of bitstream RE is to determine the configuration of a configurable device based upon analysis of the bitstream. This determination must be made starting with incomplete information about the physical device fabric and interpretation of the bitstream. The RE process must extract additional information, and any uncertainties in this extraction reduce confidence in the correctness and completeness of any resulting configuration estimate.

The estimation of device configuration should be accompanied by an estimation of the confidence that should be accorded the result. Any process leading to an estimation of confidence will depend significantly upon details of the RE process itself. Therefore, a process for confidence estimation cannot be defined fully in the abstract. However, the necessary results from confidence estimation can be described in the absence of any specific RE approach.

A. State and Configuration

The total state of a digital device is normally embodied by the values present in all memory within the device. This effort separates the total state of the device into two categories: the configuration including all switch bit values that establish parameters of logic elements and interconnections, and the initialization state including all bit values held in registers that do not directly establish logic element or interconnect parameters. For example, the bits placed in a lookup table would be configuration while the bits placed in a bitstream loader address register would be state.

The storage of configuration and state bits may be volatile or persistent. Volatile storage loses its value when power is removed from the device. Values must be loaded into volatile storage after power is applied and before the device can be used to realize the design embodied in the bitstream. Most or all bits will be volatile for SRAM-based configurable devices. Persistent storage maintains its values when power is removed from the device. Values loaded into persistent storage continue to be available when power is restored and until the values are explicitly changed. FPGAs with Flash memory allows for persistent bitstream storage.

While volatile storage does lose its values when power is removed, some volatile storage may be designed to provide default initial values when power is applied. This volatile storage will always contain the same value when power is first applied, but this initial value will be independent of any value that might have been in the storage when power was removed. Uninitialized volatile storage should be assumed to contain unknown values when power is applied.

Proposition 1: All state and configuration bits in the device should be known along with their purpose and volatility. The mechanism for establishing the value of both volatile and persistent bits should be known. Any default initial values should be known.

B. Bitstream Protocol

The raw bitstream is just a sequence of bits, but this sequence is logically divided into fields for interpretation. Thus, bits have meaning based upon their positions in the stream. If the stream is packetized, each packet will have a header or address that may be followed by data. The first step to understanding the bitstream is understanding of the bitstream protocol. Additional inputs to the device might be used to exert control over bitstream loading. For example, the logical value applied to physical device pins may establish where and how the bitstream is transferred into the device; whether through serial, parallel, or JTAG ports.

Proposition 2: The division of the raw bitstream into fields and the protocol used to organize information in the bitstream should be known. Further, any mechanism that influences the transfer path and associated format should be known.

C. Foundations for Mapping

The bitstream establishes the state and the configuration of the device. The state of the bitstream loader determines how the configuration bits will be interpreted and where they will be loaded in the device fabric. Therefore, confidence must be established in the purpose of each bitstream loader state as established by the values assigned to each loader "register" or equivalent.

The Xilinx example used registers that were written to establish modes, constraints, addresses, and other operational factors for the bitstream loader. The purpose and operation of each of these registers must become known as part of the RE process. The device manufacturer may make available information about these registers, and this information is likely to be correct but incomplete. For example, the manufacturer may provide names for most of these registers and provide additional register descriptions for some of these registers. Other registers may not be described or even identified.

Proposition 3: The purpose, format, and access of each bitstream loader state register (or equivalent) should be known. Mappings between bitstream elements (such as packets) and resulting changes to bitstream loader state registers should be known.

The bitstream loader state influences the interpretation of bitstream elements used to configure the device fabric. For example, an "address" value in the bitstream loader state may be used to indicate the starting point in the device fabric for loading bits from bitstream elements. As another example, an "order" value in the bitstream loader state might indicate the order of interpretation of data words such as big endian or little endian.

Proposition 4: The aggregate bitstream loader behavior as controlled by the bitstream loader state should be known. The bitstream loader will interpret some bitstream elements as data to be used to configure the device fabric. This data will establish logic structure within the fabric and interconnection among logic devices. Each data bit will be sent by the bitstream loader to a specific location within the fabric. This transfer effectively configures the fabric to realize a specific design.

Proposition 5: The mapping between fabric configuration data in the bitstream and specific switch element addresses in the fabric should be known.

Specific switch elements in the fabric serve specific purposes. For example, the switch element at address "A" might establish the value of the third bit in the lookup table found in logic block "B" that is part of logic block group "C" in logic block group array "D" in frame "E." This mapping provides the correspondence between the bitstream and the fabric configuration.

Proposition 6: The mapping between switch elements addresses and corresponding switch elements in the fabric should be known.

D. Configuration Complexity

With complete knowledge of device details and state, the bitstream provides a unique and deterministic device configuration. Absent or obscured knowledge introduces uncertainty into the interpretation of the bitstream to determine the device configuration.

Functional Equivalence Insufficiency: A single design can be realized in a large device using many different configurations. That is, the mapping of a design to a device is a one-to-many process. Typical toolchains used to convert design descriptions to bitstreams provide options to favor speed or area. Different design realizations may perform identical functions with very different performance, area, and possibly power requirements. Thus, a RE approach that determines functionality without additional implementation details may be insufficient for some applications.

Proposition 7: The RE approach should determine the actual configuration of the device instead of just the functionality of the device.

Significance of Unused Areas: Very few designs use all of the resources available in a configurable device. The remaining device resources are not needed for the design and may remain unused. However, every switch bit in the fabric will have some value whether the bit is needed for the design or not. The RE process should determine the configuration that exists in those unused areas to detect unnecessary power-consuming structures or malicious additions.

Proposition 8: The RE process should determine the structure of the entire device including both areas used by the design and unused areas.

Significance of Power Distribution: Large configurable devices can have complex systems of distribution providing

power to different areas of the device. The location of different parts of the design relative to this power distribution system can influence the design behavior.

Proposition 9: The RE process should determine the locations of the various design parts with respect to the power distribution system.

Significance of Clock Distribution: Large configurable devices can have complex systems of distribution providing clock signals to different areas of the device. The location of different parts of the design relative to this clock distribution system can influence the design behavior.

Proposition 10: The RE process should determine the locations of the various design parts with respect to the clock distribution system.

E. Establishing Confidence

Most of these RE objectives will be difficult to achieve because device manufacturers limit the information that they provide about their devices. In fact, limitation or denial of access to this information creates the need for RE. Fortunately, extraction of useful configuration data from the bitstream in the absence of complete information should be possible. The value of the extracted configuration will typically be influenced by the confidence in the correctness of the configuration.

Confidence in the result may be estimated by assembling evidence supporting the correctness of each RE objective. For example, the engineer might assume that information provided by the manufacturer is correct, and thus assign a perfect confidence score to this information. Other information may be extracted by performing tests on the device. Information thus extracted will depend upon analysis of the tests performed and the information exposed by those tests in aggregate. Confidence values for information obtained through tests depends upon the design of the tests and the results obtained.

Some information may not be obtainable from tests and will require assumptions. These assumptions may reduce confidence in the result, but a sensitivity analysis may be applied to determine the impact. For example, the purpose of register "A" may be completely unknown, but register "A" may never be written in a particular bitstream. While register "A" was likely included in the device for some purpose, that purpose was either not relevant to the design represented in the analyzed bitstream or a default value was used. In this case, the absence of knowledge about register "A" should reduce the confidence in the results, but any reduction may be minimal.

Assessment of confidence in results depends upon the process used to obtain those results. Thus, the assessment of confidence critically depends upon the process used to reverse engineer the bitstream. Methods to quantify the "weight of evidence" could potentially be used to assess the cumulative strength of various lines of evidence for or against a particular hypothesis or claim [30-32].

IV. Conclusions

This paper presents issues for consideration to establish confidence in the results of bitstream RE. A generic RE problem was considered without reference to any specific configurable device or RE approach. Several categories of information were identified that should be extracted by the RE process. Information in these categories is unlikely to be extracted with perfect confidence, so confidence values should be assigned to the results obtained. The confidence values for each category should be determined based upon the tests performed and information gathered through the RE process.

There should be significant opportunity to develop measures for confidence in specific bitstream engineering techniques. These measures will need to be developed in the context of those specific techniques, and the categories of information described in this paper can be used to guide confidence measure development. Future work should include the development of confidence scores in concert with the development of bitstream RE approaches and risk-based standards [33].

References

[1] Yahoo News, "China, Russia could be reverse engineering US military equipment left behind in Afghanistan: Trump," Yahoo News, [online], Available: https://sg.news.yahoo.com/china-russia-could-reverse-engineering-074456314.html. [Accessed 20 October 2021].

[2] M. Fyrbiak, S. Strauß, C. Kison, S. Wallat, M. Elson, N. Rummel, and C. Paar, "Hardware reverse engineering: Overview and open challenges," 2017 IEEE 2nd International Verification and Security Workshop (IVSW), 2017, pp. 88-94.

[3] J. Kumagai, "Chip detectives [reverse engineering]," IEEE Spectrum, vol. 37, no. 11, pp. 43-48, 2000.

[4] M. G. Rekoff, "On reverse engineering," IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-15, no. 2, pp. 244-252, 1985.

[5] E. J. Chikofsky and J. H. Cross, "Reverse engineering and design recovery: a taxonomy," IEEE Software, vol. 7, no. 1, pp. 13-17, 1990.

[6] P. Samuelson and S. Scotchmer, "The Law and Economics of Reverse Engineering," Yale Law Journal, vol. 111, no. 7, pp. 1575-1664, 2002.

[7] M. Fyrbiak, S. Wallat, P. Swierczynski, M. Hoffmann, and S. Hoppach, "HAL—The Missing Piece of the Puzzle for Hardware Reverse Engineering, Trojan Detection and Insertion," IEEE Transactions on Dependable and Secure Computing, vol. 16, no. 3, pp. 498-510, 2019.

[8] C. Bao, D. Forte, and A. Srivastava, "On Reverse Engineering-Based Hardware Trojan Detection," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 35, no. 1, pp. 49-57, 2016.

[9] S. Wallat, M. Fyrbiak, M. Schlögel, and C. Paar, "A look at the dark side of hardware reverse engineering - a case study," 2017 IEEE 2nd International Verification and Security Workshop (IVSW), pp. 95-100, 2017.

[10] W. Danesh, J. Banago, and M. Rahman, "Turning the Table: Using Reverse Engineering Techniques to Detect FPGA Trojans," in Proceedings of ACM Woodstock conference (WOODSTOCK'18). ACM, New York, NY, USA.

[11] S. Wallat, N. Albartus, S. Becker, M. Hoffmann, and M. Ender, "Highway to HAL: Open-Sourcing the First Extendable Gate-Level Netlist Reverse Engineering Framework," in Proceedings of the 16th ACM International Conference on Computing Frontiers, Alghero, Italy, 2019.

[12] D. J. Celebucki, Methods of Reverse Engineering a Bitstream for Field Programmable Gate Array Protection, Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, 2018, p. 106.

[13] 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, Tokyo, 2019.

[14] H. Yu, H. Lee, S. Lee, Y. Kim, and H.-M. Lee, "Recent Advances in FPGA Reverse Engineering," Electronics, vol. 7, no. 10, 2018.

[15] E. Bergeron, L.D. Perron, M. Feeley, and J.P. David, "Logarithmic-Time FPGA Bitstream Analysis: A Step Towards JIT Hardware Compilation," TRETS, 2011, 4.

[16] H. Yu, H. Lee, Y. Shin, and Y. Kim, “FPGA reverse engineering in Vivado design suite based on X-ray project,” In Proceedings of the 2019 International SoC Design Conference (ISOCC), 2019, 239-240.

[17] S. Choi, J. Park, and H. Yoo, “Reverse Engineering for Xilinx FPGA Chips using ISE Design Tools,” J. Integr. Circuits Syst. 2020, 6, 1.

[18] S. Choi and H. Yoo, “Fast Logic Function Extraction of LUT from Bitstream in Xilinx FPGA,” Electronics 2020, 9, 1132.

[19] T. Zhang, J. Wang, S. Guo, and Z. Chen, “A Comprehensive FPGA Reverse Engineering Tool-Chain: From Bitstream to RTL Code,” IEEE Access 2019, 7, 38379–38389.

[20] J. Yoon, Y. Seo, J. Jang, M. Cho, J. Kim, H. Kim, and T. Kwon, “A Bitstream Reverse Engineering Tool for FPGA Hardware Trojan Detection,” In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018.

[21] Y. Seo, J. Yoon, J. Jang, M. Cho, H. Kim, and T. Kwon, “Poster: Towards reverse engineering FPGA bitstreams for hardware trojan detection,” In Proceedings of the Network Distribution System Security Symposium (NDSS), San Diego, CA, USA, 18–21 February 2018.

[22] M. Jeong, J. Lee, E. Jung, Y.H. Kim, and K. Cho, “Extract LUT Logics from a Downloaded Bitstream Data in FPGA,” In Proceedings of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 27–30 May 2018.

[23] Z. Ding, Q. Wu, Y. Zhang, and L Zhu, “Deriving an NCD file from an FPGA bitstream: Methodology, architecture and evaluation,” Microprocess. Microsyst. 2013, 37, 299–312.

[24] F. Benz, A. Seffrin, and S.A. Huss, “Bil: A tool-chain for bitstream reverse-engineering,” In Proceedings of the 22nd International Conference on Field Programmable Logic and Applications (FPL), Oslo, Norway, 29–31 August 2012.

[25] J.B. Note and E. Rannaud, “From the bitstream to the netlist.” In Proceedings of the 16th international ACM/SIGDA symposium on Field programmable gate arrays, New York, NY, USA, 24–26 February 2008.

[26] J. Andina, E. de la Torre Arnanz, and M. Valdés Peña, “FPGAs: Fundamentals, Advanced Features, and Applications,” in Industrial Electronics (1st ed.), CRC Press, 2017.

[27] Xilinx, "7 Series FPGAs Configuration - User Guide," 20 Aug 2018. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug470_7Series_Config.pdf. [Accessed 24 Aug 2021].

[28] Xilinx, "Spartan-6 FPGA Configuration - User Guide," Xilinx, 22 Mar 2019. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug380.pdf. [Accessed 24 Aug 2021].

[29] "SymbiFlow Project X-Ray Xilinx 7-series Architecture Bitstream format," [Online]. Available: https://symbiflow.readthedocs.io/en/latest/prjxray/docs/architecture/bitstream_format.html. [Accessed 23 Aug 2021].

[30] Z. A. Collier, K. A. Gust, B. Gonzalez-Morales, P. Gong, M. S. Wilbanks, I. Linkov, and E. J. Perkins, "A weight of evidence assessment approach for adverse outcome pathways," Regulatory Toxicology and Pharmacology, vol. 75, pp. 46-57, 2016.

[31] I. Linkov, D. Loney, S. Cormier, F. K. Satterstrom, and T. Bridges, "Weight-of-evidence evaluation in environmental assessment: review of qualitative and quantitative approaches," Science of the Total Environment, vol. 407, pp. 5199-5205, 2009.

[32] D. Gough, "Weight of evidence: a framework for the appraisal of the quality and relevance of evidence," Research Papers in Education, vol. 22, no. 2, pp. 213-228, 2007.

[33] Z.A. Collier, I. Linkov, D. DiMase, S. Walters, M. Tehranipoor, and J.H. Lambert, “Cybersecurity standards: managing risk and creating resilience,” IEEE Computer, vol. 47, no. 9, pp. 70-76, 2014.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.