Late Breaking Results: Is Reconfigurable-based Obfuscation Secure?

Zain Ul Abideen[†], Levent Aksoy[‡], and Samuel Pagliarini[†‡]
[†]Carnegie Mellon University, PA, USA, [‡]Tallinn University of Technology, Estonia
{abideen, pagliarini}@cmu.edu, {levent.aksoy, samuel.pagliarini}@taltech.ee

***Abstract*—Reconfigurable-based obfuscation (REBO) techniques, such as eFPGA redaction, offer security against threats present in the globalized Integrated Circuit (IC) supply chain. Today, no attacks have succeeded in convincingly or fully breaking these techniques. At best, previous attacks have provided vulnerability analysis or have partially recovered a key (bitstream). This paper presents a novel attack to break the security of REBO. We propose a new attack to retrieve the design's bitstream and assess the effectiveness of the attack using the HeLLO CTF benchmarks. The success rate of our attack is between 57% and 62%, superseding all previous known results on these benchmarks.**

***Index Terms*—Reconfigurable-based obfuscation, Secure ASIC Design, Fine Grain Redaction, LUT-based Obfuscation, Logic Locking**

I. Introduction

The semiconductor industry depends on advanced technology nodes for deploying its high-performance Integrated Circuits (ICs), often outsourcing fabrication to a globalized supply chain. This outsourcing raises threats such as Intellectual Property (IP) piracy, reverse engineering, overproduction, and malicious logic insertion. To address these threats, researchers have suggested countermeasures like Logic Locking (LL), IC Camouflaging, and Split Manufacturing [1]. While LL obfuscation has been well-studied, it remains vulnerable to various attacks. Recent research has focused on understanding the *pitfalls* of LL and its countermeasures.

Recently, alternative obfuscation techniques using reconfigurable logic, such as Look-up Tables (LUTs) and embedded Field Programmable Gate Arrays (eFPGA), have emerged, with *the bitstream acting as a key* to unlock the design [2]–[5]. While promising, these techniques often incur significant Performance-Power-Area (PPA) overhead, therefore researchers typically redact only the most sensitive parts of a circuit, keeping the overheads minimal. Yet, conducting a security analysis of REbo *remains imperative*. REBO demonstrates initial resilience to SAT attacks due to the substantial length of its bitstream, which can extend to 10^4 or even 10^6 bits. This significant bitstream length makes it secure against such attacks[1]. For this reason, designs based on this technique were included in the Hello:CTF 2023 competition sponsored by Intel [6].

Considering proposed attacks on REBO, it is worth highlight that "Break & Unroll attack" that consists of two phases - cycle breaking and unrolling [7]. Another type of attack, described in [8], is the "Predictive Model Attack". This attack uses machine learning techniques to construct a predictive model that replicates the original circuit's behavior, which can then be exploited for the attack. Another recent attack, "FuncTeller", utilizes modern EDA tools to locate minterms of interest to query the circuit [9].

The attacks mentioned earlier have not entirely compromised the obfuscation techniques based on reconfigurability. Instead, attacks have either recovered the key for a small circuit or reported vulnerabilities. It is important to note that the selected benchmarks and designs used in the attacks have only a small number of keys. The authors of [7] have analyzed the design with a maximum bitstream size of 4000 bits, while [8] considered designs with signal processing applications. The authors of [9] evaluated the dated ISCAS'85 benchmarks and a few medium-sized designs, such as MIPS processors.

II. Attack Methodology

In this paper, we propose a novel attack to evaluate the security of REBO techniques. The steps involved in the proposed attack are visually represented in Fig. 1. REBO exploits LUTs to hide the logic behind a bitstream. We have identified the untrusted foundry as the primary adversary in our threat model. Whether the adversary is an institutional entity or a rogue employee, we do not differentiate. Furthermore, we consider the following assumption: 1) An adversary with access to the design's GDSII file can easily understand the layout representation. 2) The adversary can access the oracle and distinguish between the reconfigurable logic (i.e., eFPGA) and non-reconfigurable logic. 3) The layout can be reverse-engineered back to a gate-level netlist. 4) The adversary can convert standard cells in non-reconfigurable logic back to LUT format and enumerate all LUTs and their input keys.

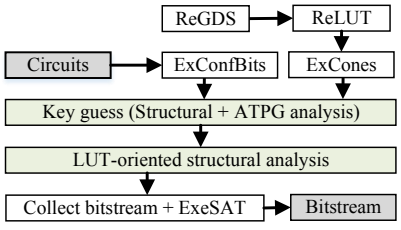

Fig. 1: The proposed attack, along with the internal steps involved

The attack methodology consists of two parts. Fig. 2 illustrates the **first part**: *ReGDS* performs reverse engineering of the layout and converts it to a gate-level netlist [10]. Our algorithm, *ReLUT*, performs reverse-engineering by grouping gates and transforming the design into LUT representation. This step utilizes structural templates to identify n-input LUTs. This also

L. Aksoy acknowledges co-funding by the European Union and Estonian Research Council via project TEM-TA138.

[1]The complexity of the SAT attack exceeds current computational capabilities when a large bitstream is involved.

978-3-9826741-0-0/DATE25/© 2025 EDAA

determines the maximum LUT size used in the obfuscated design[2]. The algorithm searches for different template-driven structures. For example, a 2-input LUT could have more than one structure. Then, the *ExConfBits* step extracts the configuration bits of every LUT instance from 50 known circuits, for example, 2-input, 3-input, 4-input, etc. The extracted unique configuration bits will be used further. After presenting the design in LUTs, the *ExCones* step extracts the circuit cones, followed by the path from the input to the FFs. Every logic cone starts from the inputs and ends at the output of a FF.

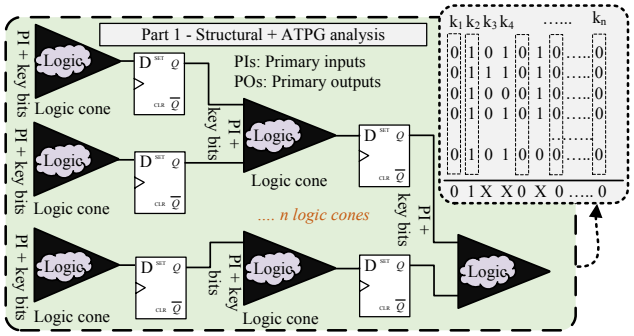

Fig. 2: Part 1: Structural and ATPG analysis in the attack

Then, the Automatic Test Pattern Generation (ATPG) analysis observes the stuck-at faults at the output of the FFs, using previous LUT configuration bits to identify test patterns (key bits) that propagate the fault from inputs to FF outputs. The goal is to achieve excellent fault coverage with minimum patterns and execution time. After finding all the test patterns (key bits), the stable bits are identified while the remaining bits are represented as unknown bits "X".

In the **second part** of the attack, we analyze the structural aspects from the LUT perspective, focusing on LUT types, appearances, and unique configuration bits, as shown in Fig. 3. The *ExConfBits* step revealed that many configuration bits are repeated. Utilizing this, we created an internal database using the most frequent and unique bits. The benchmarks involve 4-input LUTs, resulting in a database of 6257 configuration bits from 4-input and 198[3] from 3-input LUTs, based on the 50 known circuits. This narrows the attack's search space to 2^{16} and 2^{8}, respectively. We apply a Pearson coefficient to correlate the design's frequency and structure with the database, yielding partial key bits as a bitstream for the targeted design, while other bits remain "X".

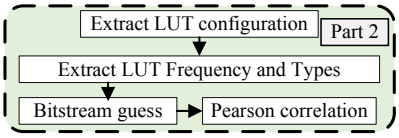

Fig. 3: Part 2: LUT-oriented structural analysis in the attack

The bitstreams collected from the *first part* and the *second part* are merged by selecting common bits from a set of bitstreams. Another level of the search space for the bitstream is now reduced further. So far, the attack does not require an oracle and can identify many key bits. Next, the oracle is used in conjunction with a SAT attack to determine the key bits of the bitstream. The SAT attack employs the targeted set of bitstreams discovered in previous steps. This completes the goal of the attack to find the highest percentage of key bits in the bitstream.

A. Results

We selected the benchmarks from the HeLLO CTF [6] contest for our security evaluation. The benchmarks consist of three instances: *low*, *medium* and *high*, which are beyond the capabilities of known attacks. Table I lists the results with the circuit characteristics, including number of inputs, outputs, gates, FFs, bitstream length, correct key bits in the bitstream, and attack execution time in seconds.

To summarize, we have made **significant** progress in retrieving the key of the obfuscated designs. Specifically, in the case of the *low* design, we generated 83 logic cones for the primary outputs. Through this method, we determined the values of 953 key bits out of a total of 1664. We conducted partially successful attacks on all instances within a reasonable amount of time. The overall success rate for all three circuits was 57%, 62%, and 56%, respectively. The execution time includes only the oracle-less portion. The total execution time is the CPU time shown in the final column of Tab. I plus an additional 48 hours for the SAT attack.

Circuit	#in	#out	#gates	#FFs	#bitstream	#bitstream (c)	CPU (s)
low	21	5	973	74	1664	953	9.6
medium	38	320	7062	1440	16080	10076	109.8
high	39	97	33642	1314	76408	42746	6944.4

TABLE I: Attack results for the HeLLO: CTF benchmarks

III. Conclusion

A new attack on REBO designs has been developed to assess the effectiveness of their security. Our attack considers designs of multiple sizes, from small to large, that are much larger than those previously analyzed in other works. The key recovery rate of our attack, even if not perfect, is significant when considering designs that have bitstreams with several thousand bits. Finally, our attack can be helpful for first-order vulnerability analysis given its probabilistic nature.

[2]HeLLo CTF benchmarks employ LUTs of up to 4-inputs.

[3]There are only 16 possible configurations for a 2-input LUT.

References

[1] M. Yasin *et al.*, *Trustworthy Hardware Design: Combinational Logic Locking Techniques*. Springer, Cham, 2019.

[2] Z. U. Abideen *et al.*, "A security-aware and LUT-based CAD flow for the physical synthesis of hASICs," *IEEE TCAD*, pp. 1–1, 2023.

[3] J. Bhandari *et al.*, "Exploring eFPGA-based redaction for IP protection," in *2021 IEEE/ACM ICCAD*, 2021, pp. 1–9.

[4] C. M. Tomajoli *et al.*, "ALICE: an automatic design flow for eFPGA redaction," in *Proceedings of the 59th ACM/IEEE DAC*, 2022, p. 781–786.

[5] Z. U. Abideen *et al.*, "An overview of fpga-inspired obfuscation techniques," *ACM Computing Surveys*, vol. 56, no. 12, pp. 1–35, 2024.

[6] B. Swarup *et al.*, "HELLO: CTF 2023," (last accessed on Feb 05, 2024. [Online]. Available: https://helloctf.org/23/rules/

[7] A. Rezaei *et al.*, "Evaluating the security of eFPGA-based redaction algorithms," in *Proc. of the 41st IEEE/ACM ICCAD*, 2022, pp. 1–7.

[8] P. Chowdhury *et al.*, "Predictive model attack for embedded FPGA logic locking," in *Proceedings of the ACM/IEEE ISLPED*, 2022, pp. 1–6.

[9] Z. Han *et al.*, "FuncTeller: How well does eFPGA hide functionality?" in *32nd USENIX Security Symposium*, Aug. 2023, pp. 5809–5826.

[10] R. S. Rajarathnam *et al.*, "Regds: A reverse engineering framework from gdsii to gate-level netlist," in *2020 IEEE HOST*, 2020, pp. 154–163.