Reconfigurable Logic for Hardware IP Protection: Opportunities and Challenges

(Invited Paper)

Luca Collini
New York University
New York, NY, USA

Benjamin Tan
University of Calgary
Calgary, Alberta, Canada

Christian Pilato
Politecnico di Milano
Milan, Italy

Ramesh Karri
New York University
New York, NY, USA

ABSTRACT

Protecting the intellectual property (IP) of integrated circuit (IC) design is becoming a significant concern of fab-less semiconductor design houses. Malicious actors can access the chip design at any stage, reverse engineer the functionality, and create illegal copies. On the one hand, defenders are crafting more and more solutions to hide the critical portions of the circuit. On the other hand, attackers are designing more and more powerful tools to extract useful information from the design and reverse engineer the functionality, especially when they can get access to working chips. In this context, the use of custom reconfigurable fabrics has recently been investigated for hardware IP protection. This paper will discuss recent trends in hardware obfuscation with embedded FPGAs, focusing also on the open challenges that must be necessarily addressed for making this solution viable.

1 INTRODUCTION

Interest in hardware intellectual property (IP) protection is increasing as governments now consider integrated circuits (IC) as critical goods for national and economic security [16]. While macroeconomic regions are developing plans to increase their manufacturing capabilities in the next decade [2], the present-day supply chain is still distributed across multiple companies across different countries, thus exposing IPs to established security threats: overproduction, counterfeiting, and malicious modifications [28]. A common denominator of these threats is reverse engineering. For this reason, many techniques for IP protection work by increasing the complexity of IC reverse engineering.

A well-known technique to protect against reverse engineering is logic locking. First proposed in [29], logic locking is a family of obfuscation techniques that works by introducing additional logic to the design. Such logic is dependent on and controlled by a new *locking key* input. The obfuscated design will work with the correct functionality if, and only if, the correct key is provided. In this way, the design can be sent for fabrication, test, and assembly, without exposing the key. The key will be loaded into a tamper-proof memory only when the chip returns to the design house. Logic locking has been proposed at different abstraction levels ranging from gate level [29, 39] up to register-transfer level (RTL) [13, 25] and even high-level synthesis (HLS) [23, 27, 40]. Logic locking introduces the concept of "equally-plausible" configurations to hide the correct one with others that the attacker cannot rule out without having any prior knowledge about the design. With an N-bit key, it aims at creating 2^n possible IC "configurations". In [8], the authors propose an interesting approach for hardware IP protection that replaces portions of the combinational logic with look-up tables based on the same concepts.

A Field-Programmable Gate Array (FPGA) is a programmable circuit that can be configured to implement different functionalities through a configuration binary called the *bitstream*. Designers can change the functionality by simply loading a new bitstream. FPGAs are usually manufactured on a dedicated chip and are sometimes paired with a micro-controller on printed circuit boards (PCBs). Embedded FPGAs (eFPGA) are FPGAs that are physically embedded within the same chip alongside other components or IP blocks. Embedding the FPGA into the same chip can bring significant benefits in terms of power, performance, and costs.

When manufacturing a chip with an embedded FPGA, only the final user knows how the eFPGA will be programmed. Thus, designers can use this approach for IP protection. They can select parts of the design and replace them with an eFPGA. During manufacturing, the attacker would not have any idea of the functionality that will be implemented, while the correct one can be simply programmed into the eFPGA before its use. This technique is called *eFPGA redaction* [11, 17, 22]. An example is shown in Figure 1.

eFPGA redaction takes the notion of equally-plausible configurations of logic locking to an extreme by completely replacing the logic to be protected with an eFPGA of which the correct configuration is known only by the designers. In this case, the "secret" key is the bitstream that configures the eFPGA. This technique is generally applied at RTL as the eFPGA tools take as input RTL modules to synthesize and create the eFPGA. For example, in [14], authors propose a redaction flow starting from HLS. Embedding an FPGA netlist into an ASIC design introduces high overheads in terms of area, power, and delay. For this reason, eFPGA redaction is only applied to sensible portions of the design, keeping the rest of the logic intact. In [6], the use of mix of static ASIC cells and configurable interconnects, called eASIC, is proposed to reduce the overhead and allow the obfuscation of bigger designs. In practice, eASIC offers a good security level only when the obfuscation is close to 100% which is when the design is fully reconfigurable, similarly to an FPGA design. Another approach for reducing the overheads was investigated by [33] using Field Programmable Transistor arrays (FPTA) instead of FPGA. While FPTAs promise less overhead, all

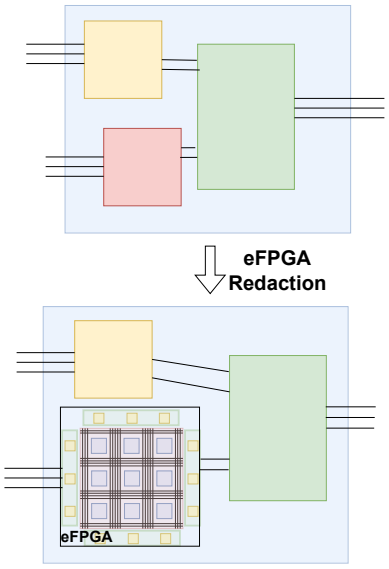

Figure 1: eFPGA redaction removes a module from the original design and substitutes it with an embedded FPGA that will be programmed after fabrication.

other challenges are the same, while the support for FTPA is not as mature as the FPGA.

Designers need to choose FPGA fabric parameters and the modules to be redacted. These decisions play a significant role when it comes to overhead and security. To provide insights into the challenges and research opportunities for eFPGA redaction, this paper is organized as follows.

Section 2 introduces the threat models and attacks against eFPGA redaction that are usually considered in the literature. Section 3 presents the main challenges that designers need to face when applying eFPGA redaction. We show the fabric provider alternatives, including both commercial and open source solutions, how fabric parameters affect both security and overhead, the challenge of choosing the design portions to be redacted, and the final integration challenges. Section 4 presents an analysis of the works that focused on applying eFPGA redaction. Section 5 reports the experience of the CSAW Logic Locking Conquest in 2021, when the challenge was focused on eFPGA redaction. We discuss the best attack approaches identified by the red teams. Section 6 summarizes the open research opportunities in the field, highlighting potential future directions of work.

2 THREAT MODELS AND ATTACKS

2.1 Threat Models

The threat models considered for eFPGA redaction are the classical threat models considered also for hardware obfuscation and, in particular, for logic locking. They revolve around the concept of an *oracle*. In this context, an oracle is a chip that performs the correct functionality[30]. The scan chain also plays a role in the threat models, especially for FPGA designs. In particular, the scan chain is a testing interface that allows one to read and write values into registers. In the case of eFPGA, scan chains could be used to inspect I/O pins of the reconfigurable fabric.

In an **oracle-less threat model**, we assume that attackers have access only to the obfuscated netlist. A malicious foundry can obtain the obfuscated netlist by reverse engineering the GDSII layout files. This is a plausible threat model for low-volume applications where it is safe to assume that attackers cannot access a working chip.

In an **oracle-guided threat model**, we assume that the attackers can access an oracle chip in addition to the obfuscated netlist. The oracle can have full, partial, or no scan chain access. This threat model is plausible for all mass-produced chips for which we can assume that attackers can access working chips to analyze the I/O relationships. Depending on the presence of the scan access, the complexity of the attacks varies a lot. Full scan access allows the attackers to query input/output pairs at each combinational logic cone, reducing the problem into many smaller problems. When full scan access is not available, the obfuscated circuit becomes sequential. Attacks on sequential circuits are more complex. Scan access could be limited by physically connecting fewer registers to the scan chain or by the use of scan-limiting techniques such as [21]. A similar approach in [20] fortifies RTL locking.

2.2 Oracle-less Attacks

Many oracle-less attacks have been proposed for logic locking. These attacks can only infer information from the obfuscated netlist. Modern attacks are based on machine learning [35] and require logic locking methods to create "balanced" designs that avoid structural biases which are used to recover the correct functionality [34]. Since eFPGA redaction completely removes the logic that is being obfuscated and replaces it with generic configurable blocks, the only part of the design that can leak information is around the eFPGA. To the best of our knowledge, there are not yet any oracle-less attacks specifically aimed at eFPGA redaction.

2.3 Oracle-guided Attacks

In the oracle-guided threat model, eFPGA redaction has been first evaluated against state-of-the-art SAT attacks, a powerful attack against logic locking [36]. It works by identifying distinguishing input patterns (DIP) through an SAT solver. DIPs are input patterns for which at least two keys will yield a different value. This is powerful because giving as input a DIP to the oracle allows one to rule out incorrect keys. In most cases, applying a DIP allows attackers to rule out several keys because many keys can be equivalent to each other and thus ruled out simultaneously [36]. Many variations of the SAT attack have been proposed through the years. The most relevant are the attacks that can handle also the combinational cycles, which are present in eFPGA fabrics. For example, CycSAT [41] solves the problem of cycles by assuming that the correct circuit will not have combinational cycles and adds constraints to the SAT solver to guarantee this condition. BeSAT [32] works by pruning out the keys that create stateful cycles. IcySAT [31] identifies a subset of feedback nets that make the netlist acyclic when removed. It then unrolls the netlist on those nets to obtain an acyclic circuit on which a classic SAT attack can be executed.

These attacks require full scan chain access. If scan chain access is limited, it may be possible to use the technique proposed in [18]

to perform SAT attacks on sequential circuits. The technique works by unrolling sequential loops to obtain a combinational netlist. Since unrolling an eFPGA fabric significantly increases the amount of logic to be analyzed, it is likely that this attack would become unfeasible. However, to the best of our knowledge, this problem has not been investigated in detail.

A novel attack specifically aimed at breaking eFPGA redaction was proposed in [15]. The attack works by sampling input, output, and latency triplets from the oracle netlist on some samples and then using this information to create a function that would fit on the eFPGA fabric. On one hand, this attack requires scan access only to the interface of the eFPGA and not to its internal registers. On the other hand, this approach allows the attacker to retrieve only a partial functionality since it is unpractical to query an exhaustive number of samples.

3 CHALLENGES

3.1 Commercial vs. Open-Source (e)FPGAs

To date, there are limited options for eFPGA fabrics, so the first challenge is to identify a suitable fabric provider and the associated tool flow. It is also important to understand whether the FPGA fabric can be customized, i.e., it is possible to create instances by varying the FPGA fabric parameters (e.g., number of inputs and outputs of the configurable blocks, size of them multiplexers, etc.) Looking at commercial solutions for eFPGA fabrics, we identified Flex logix [3], Menta eFPGA [4], Achronix [1], and QuickLogic [5]. For open-source solutions, we can find OpenFPGA [37] and FABulous [19].

Flex Logix aims for high-performance fabrics for large IPs, where the smallest building block counts over a thousand LUTs. So, it is not feasible to redact small portions of the designs. Menta eFPGA offers solutions that range from hundreds to tens of thousands of LUT, along with optional SRAMs and DSPs. The company mentions IP security among its objectives. Achronix provides both traditional and embedded FPGAs. The architecture of the basic blocks cannot be customized. They seem to offer from 10k LUTs to millions of LUTs solutions, which do not seem practical for eFPGA redaction. QuickLogic offers an eFPGA IP generator that is based on the open-source OpenFPGA project. Rapid Silicon also builds upon OpenFPGA, offering a system-on-chip with eFPGA fabric and an EDA environment for development.

OpenFPGA is a mature framework for rapidly prototyping customizable FPGA architectures [37]. OpenFPGA allows the customization of the FPGA fabric, where the parameters are specified through an XML file. It can also be used for eFPGAs and eFPGA redaction, as shown in [11]. OpenFPGA was not initially created for eFPGAs. So it requires additional steps to integrate the newly-created FPGA into the original design. However, this process can be automated, as shown in [38] for eFPGA redaction.

FABulous [19] is a novel framework that simplifies the use of eFPGAs. It also allows for the customization of eFPGA fabrics and covers the whole flow. Since it is in early development stages, the documentation lacks some details and the flow does not appear to be fully automated yet. Table 1 summarizes the characteristics of the different eFPGA providers.

Table 1: Summary of eFPGA providers

Tool	LUT #	Customizable LB Architecture	DS-blocks
Flex Logix	1k-500k	✗	MAC, BRAM
Menta eFPGA	100-200k	✗	DSP, RAM
Achronix	1k-1m	✗	DSP, MLP, LRAM, BRAM
QuickLogic	*	✓	DSP, BRAM
Rapid Silicon	*	*	
OpenFPGA	100-1m	✓	DSP, Memory
FABulous	100-1m	✓	

* details were not clear to the authors at the time of writing

3.2 eFPGA Fabric Configuration

Another important challenge is the selection of the fabric configuration. Indeed, the choice of the eFPGA fabric parameters plays a major role in terms of security, as they determine the complexity of the fabric, the "capacity" of the fabric (i.e., the scope of functionality that the fabric can implement), and its bitstream length. For example, OpenFPGA offers a tile-based architecture composed of I/O tiles, Connection Blocks, and Configurable Logic Blocks (CLB). The tiles are arranged into a grid. The outer layer is composed of I/O tiles with missing tiles in the angles. Figure 2 illustrates the tile architecture of OpenFPGA.

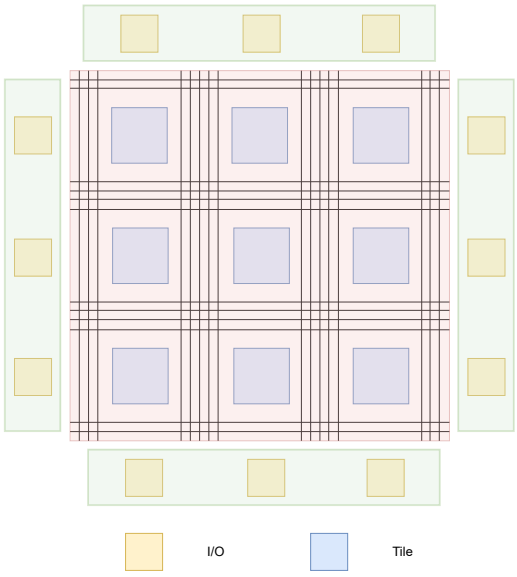

Figure 2: Tile architecture of of a 5x5 FPGA from OpenFPGA

A CLB is composed of N Basic Logic Elements (BLE), each of them implementing a primitive logic function through a Look-Up Table (LUT), a Flip-Flop (FF), and a 2-input multiplexer. The numbers of BLEs (N), the size of LUTs (K), and the number of inputs to the CLB (I) are parameters that can be specified to customize the fabric architecture. Figure 3 shows a simplified scheme of a CLB and BLE from OpenFPGA [10, 37].

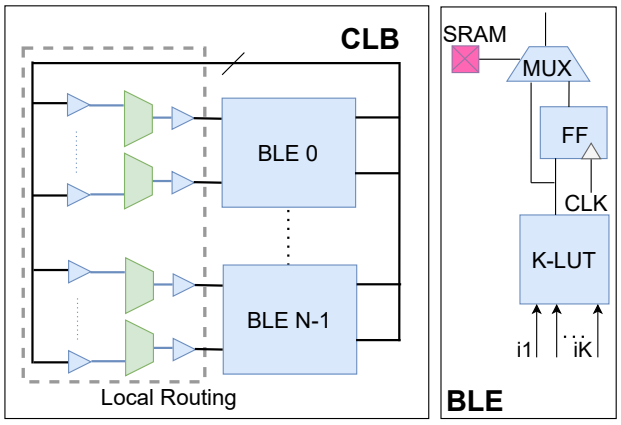

Figure 3: Simplified architectures of Configurable Logic Block and Basic Logic Element.

3.3 eFPGA Parameters and Security

The different parameters determine the programmability of the fabric and, thus, the overall bitstream size. Note that only considering the bitstream (key) size is not appropriate as a security metric (in the context of an oracle-guided attack). Indeed, as shown in [10], the different programmable parts of a fabric (e.g., routing configuration, LUT contents) affect SAT attack resistance in different ways. In the same work, the authors studied how security and overheads change when varying N and K, while I was defined as $I = \frac{K(N+1)}{2}$ since this value was able to yield good Power, Performance, Area (PPA) metrics [7]. This work also shows that N (i.e., the number of BLEs) increases the complexity of the cyclic networks, hardening SAT resilience more than K (e.g., the LUT size). It must be noted that K also increases SAT resilience. Another important is that also resource utilization of the eFPGA fabric plays a role in terms of security. Indeed, since the fabric is customized for the specific modules to be redacted, eFPGAs with low utilization will contain large portions of the logic that are optimized. This effect may result in a simplification of the corresponding SAT attacks. On the contrary, highly-utilized eFPGAs will result in large bitstreams to be recovered without any simplification that may help the attacker. The existing study only focused on the parameters N and K, leaving unexplored the effects of the parameters for connecting blocks and global routing.

3.4 Module Selection

When applying eFPGA redaction, it is important to carefully select which module(s) should be "moved" to the eFPGA based not only on the designer's security objectives but also on the PPA effects. In the context of naive oracle-guided SAT attacks, the nature of the redacted module was shown not to play a large role in terms of resistance [11]. This key observation allows designers to freely choose the parts of the IC to be redacted. However, most of the times, it is not possible to redact the entire IP module as the resulting overhead would be too high. In addition, some parts of the IP may not be worth being protected as they implement public or well-known functions (i.e., I/O management and synchronization). ALICE [38] is a framework proposed to address this challenge. It automatically identifies a set of modules that satisfy overhead requirements while maximizing I/O utilization. To make sure that the redacted modules are worth being protected, ALICE requires the designers to specify which outputs are relevant to be "obfuscated". It then identifies which modules affect those signals and flags them as *security relevant*.

3.5 eFPGA Integration

Once the modules to be redacted are defined and the corresponding eFPGA fabric is created, designers need to integrate it with the rest of the chip, routing to/from it the signals used by the redacted modules. The integration of the obtained eFPGA varies depending on the tool used to create the eFPGA. When using an eFPGA-specific tool like FABulous, this process is done by the tool. This also means that the designer has limited control over this process. When using a more generic FPGA customization tool (e.g., OpenFPGA), this process is left to the designer. Depending on the number of redacted submodules, this process can offer different alternatives. In general, the eFPGA is instantiated in the top module and signals must be rerouted. Using clever approaches, like the one proposed in [38] that is based on the concept of dominator trees, reduces the signal propagation and, in turn, routing issues. The same approach has been used in HLS to allocate internal memories closer to the modules where they are accessed [26]. However, the trade-off between security and overhead effects has not been investigated yet.

3.6 Security Validation

There have been many attacks for logic locking techniques [12]. While researchers started to formalize notions of logic locking, including redaction, only recently [9], scrutiny of eFPGA redaction is still in its infancy. Currently, the security of eFPGA redacted modules is evaluated with the same approaches used for logic locking, aiming at recovering the bitstream especially with SAT attacks. Thus, a definitive evaluation of the security of a given integration between module and fabric remains an open challenge. Progress towards this challenge would also foster the development of design space exploration to trade off security and integration costs (i.e., the added resources needed for the fabric, impacts on timing, etc.).

4 EXISTING REDACTION APPROACHES

In [17], authors explored eFPGA redaction using a predefined and fixed eFPGA. This is compatible to taking an eFPGA from a vendor listed in Section 3.1. This approach showed the feasibility of an HLS flow supporting pragmas to select the portions of designs to be redacted on an eFPGA provided by the designer. The limitations of this work were in the manual work for the designer to select the parts to be redacted and provide a proper eFPGA fabric.

In [22], the authors advanced the work in eFPGA redaction by providing a tool based on Chisel HDL to generate a custom eFPGA fabric fitted for the redacted logic. Another work proposed the use of OpenFPGA to obtain the fitting FPGA to be used for eFPGA redaction [11] . Both works evaluated the security of the obfuscated solutions applying the state-of-the-art cyclic SAT attacks mentioned in Section 2, showing good resilience. Common challenges for both

Table 2: Summary of eFPGA architecture provided

instance	#clbs	#clbluts	#lutins	#luts	instance	#clbs	#clbluts	#lutins	#luts
set1_1	4	2	3	8	set3_1	4	2	5	8
set1_2	4	3	3	12	set3_2	4	3	5	12
set1_3	4	4	3	16	set3_3	4	4	5	16
set1_4	4	5	3	20	set3_4	4	5	5	20
set1_5	4	6	3	24	set3_5	4	6	5	24
set1_6	4	7	3	28	set3_6	4	7	5	28
set1_7	4	8	3	32	set3_7	4	8	5	32
set2_1	4	2	4	8	set4_1	4	2	6	8
set2_2	4	3	4	12	set4_2	4	3	6	12
set2_3	4	4	4	16	set4_3	4	4	6	16
set2_4	4	5	4	20	set4_4	4	5	6	20
set2_5	4	6	4	24	set4_5	4	6	6	24
set2_6	4	7	4	28	set4_6	4	7	6	28
set2_7	4	8	4	32	set4_7	4	8	6	32

works included the choice of the logic to be redacted to properly fit onto an eFPGA which respects overhead constraints.

HLock [14] is a framework to apply eFPGA redaction during HLS. This flow automatically identifies the security relevant parts of the design from an IP perspective. It works by taking two high-level descriptions of the same design, one with a standard implementation (HDL_{known}) and the other with an improved implementation (HDL_{new}). The claim is that the difference between these two designs is the security relevant logic from an IP perspective. The framework synthesizes the two design with HLS and proceed by extracting the Data Flow Graph (DFG) of HDL_{known} and HDL_{new}. It then finds the maximum subgraph in common between the two DFGs. The logic corresponding to this subgraph is considered as not security relevant as it is available in known implementations. The remaining logic is considered security relevant and is mapped onto an FPGA. The framework was not evaluated with an actual eFPGA, but instead traditional, off-the-shelf FPGAs are used for redaction. One major limitation of this work is that it requires two different implementations of the same design to identify the security-relevant parts. This situation is however uncommon for new IC designs.

ALICE [38] proposes an end-to-end flow to analyze a single design at RTL and identify which modules affect selected outputs. These modules are candidates for eFPGA redaction. ALICE includes also a cluster and pruning step to place more independent modules onto the same eFPGA and to eliminate unfeasible implementations (e.g., modules that require more I/O pins that the ones allowed by the designer). This approach also performs an automatic integration of the eFPGA fabric (generated with OpenFPGA) and the rest of the chip. the major limitation is that the eFPGA parameters (and so the security level of the eFPGA) are given and not co-explored with the modules to be redacted.

In conclusion, all approaches focus on specific aspects of the eFPGA redaction problem, while an holistic approach that optimizes both security and EDA aspects is still missing.

5 A RED-BLUE TEAM EXERCISE: LESSONS

5.1 Scenario

To encourage more engagement and scrutiny of eFPGA redaction, the CSAW 2021 Logic Locking Conquest [24] provided the opportunity for a Red-Blue team exercise. Red teams were given a set of modules redacted in eFPGA fabrics and were asked to recover the original functionality under the oracle-guided attack model. The redacted design (a simple adder) was kept common and the different fabrics varied in terms of LUT number and sizes in the CLBs. The competition was divided into two phases, the first one spanning from August to October and the second one lasting two weeks from the second half of October to the early November. In the second phase, additional, more complicated fabrics were provided to the red teams. Table 2 reports the fabric characteristics of each design. The table reports the size of the fabric along with the number of LUTs and other configurable blocks.

Proposed attack approaches ranged from applying existing SAT techniques and through to interesting variations or completely new attacks to retrieve the correct bitstream. We now discuss our observations of the attacks proposed by the participants.

5.2 Experiences

One team came up with an interesting new functional recovery attack exploiting some knowledge on the distribution of the Primary Implicants (PI) in the Primary Implicants Table (PIT). They noted that in many circuits, including the redacted one, the PI are close together in the PIT. They exploited this by first finding a PI by performing a set of queries and then searching for PIs close-by. Once all PIs are identified they can synthetize the functionality for the eFPGA architecture at hand and obtain a working bitstream They implemented the attack and showed an average accuracy greater than 99%.

Another team came up with a different approach, based on the idea that the eFPGA is a universal circuit and can be substituted with a simpler but equivalent universal circuit to recover functionality. The first step of their attack was to identify the "logic capacity" of the eFPGA fabric and create a simple universal circuit with the same capacity. Then they launched a SAT attack on the generated circuit and once the functionality is retrieved they can resynthesize it for the architecture of the eFPGA obtaining the unlocking bitstream. The team implemented the attack which was only partially successful due to some fabrics being big enough to make the attack fall into a brute force attack.

Some teams tried existing SAT attacks and proposed variations to circumvent the challenges in applying SAT attacks to eFPGA fabrics. The main problem when applying SAT attacks to eFPGA fabrics is the presence of combinational cycles introduced by the reconfigurable routing network. In some instances these cycles can be resolved by adding constraints to brake the loops (CycSAT) and SAT attacks can be performed. When the eFPGA fabrics often present some intertwined loops that are hard to break and lead to exponential complexity.

Other teams tried existing publicly available implementations of SAT attacks and were able to break only a limited number of designs. This was due to the lack of out-of-the-box support for the combinational loops present in the eFPGA fabrics. One team tried to modify the SAT attack in an incremental manner to break portions of the design at a time, but the technique did not prove effective.

5.3 Lessons Learned

The big obstacle in breaking eFPGA fabrics proved to be the presence of hard cycles. In fact the most successful teams were the ones that managed to circumvent the hard cycles by focusing on the functionality itself. Even though these attacks have a more limited scope (require knowledge on the PI distribution) it highlights how "just making the fabric more complex" could not be the solutions to all problems. Oracle-less attacks were not successful in the competition as the red teams were provided with only the eFPGA fabric without the surrounding logic of the final IC. Some teams rightfully speculated that when integrated in a bigger design some information about the surroundings could allow further leverage for attacks, so this is open for future exploration. Results showed that almost all finalists were able to break the bitstreams of the first phase designs, while the second phase designs, which included more hard cycles, were only partially retrieved from the first team. This indicates that the presence of hard cycles drastically increases the attack complexity in accordance with the findings in [10].

6 OPPORTUNITIES

An interesting research direction that remains open is the multi-objective optimization to explore different combinations of fabric parameters and number of redacted modules. In general the more secure eFPGA fabric parameters introduce higher overheads. If we put ourselves in the perspective of a hardware designer, we may be willing to spend up to a pre-fixed overhead to secure our IP. In this scenario we may be face a situation where we could have a very strong fabric that allows us to redact a small portion of logic, or some weaker fabric that allows us to redact a bigger portion of logic. Design space exploration could help us in this scenario by identifying a sweet spot between fabric parameters and logic capabilities.

The novel attacks (see Section 5.2) proposed by red teams at CSAW Locig Locking Conquest in 2021 showed interesting takes on new eFPGA-specific attacks and we expect to see further achievements from the respective research groups. It was also shown that there might be a space for SAT-based attacks but the problem of hard cycles would need to be resolved. The space for oracle-less attacks is also still open even though the challenge complexities raise drastically. As eFPGA redaction removes all the information of the redacted logic from the design, oracle less attacks need to rely on the sorrunding logic and on the logic capacity of the eFPGA fabric. Guidelines on how not to leak information when using eFPGA redaction for a provably secure scheme at least in the oracle less threat model would be very precious for the community.

It has been shown that the fabric structure affects the resilience towards oracle guided attacks, though many architectural choices remain unexplored. With the rise of FABulous it would be interesting to evaluate its use for eFPGA redaction and what architectural choices can improve the security.

7 CONCLUSIONS

This paper discusses the recent trend for hardware IP protection that involves reconfigurable devices to hide critical parts of a design. This approach, called *eFPGA redaction*, can be considered a natural evolution of logic locking. We discuss the common challenges for this problem, including the selection of the (e)FPGA provider, the potential customization of the reconfigurable fabric, and the security/EDA implications. We also discussed an experience, the CSAW 2021 logic locking contest, where groups were requested to break eFPGA redacted modules.

ACKNOWLEDGMENTS

The authors would like to thank Jitendra Bhandari and Abdul Khader Thalakkattu Moosa, the student leads of CSAW LLC 2021, and all the participants of the contest.

REFERENCES

[1] 2022. Achronix Semiconductor Corporation. https://www.achronix.com/speedcore-architecture
[2] 2022. Digital sovereignty: Commission proposes Chips Act. https://ec.europa.eu/commission/presscorner/detail/en/ip_22_729
[3] 2022. Flex Logix. https://www.flex-logix.com/efpga/what-is-efpga.html
[4] 2022. Menta eFPGA. https://www.menta-efpga.com/efpga-ip-cores-v5
[5] 2022. QuickLogic Corporation eFPGA IP 2.0. https://www.quicklogic.com/products/efpga/efpga-ip2/
[6] Zain Ul Abideen, Tiago Diadami Perez, and Samuel Pagliarini. 2021. From FPGAs to Obfuscated eASICs: Design and Security Trade-offs. In *2021 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. 1–4. https://doi.org/10.1109/AsianHOST53231.2021.9699758
[7] Elias Ahmed and Jonathan Rose. 2004. The effect of LUT and cluster size on deep-submicron FPGA performance and density. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 12, 3 (March 2004), 288–298. https://doi.org/10.1109/TVLSI.2004.824300 Conference Name: IEEE Transactions on Very Large Scale Integration (VLSI) Systems.
[8] Alex Baumgarten, Akhilesh Tyagi, and Joseph Zambreno. 2010. Preventing IC Piracy Using Reconfigurable Logic Barriers. *IEEE Design & Test of Computers* 27, 1 (Jan. 2010), 66–75. https://doi.org/10.1109/MDT.2010.24 Conference Name: IEEE Design & Test of Computers.
[9] Peter Beerel, Marios Georgiou, Ben Hamlin, Alex J. Malozemoff, and Pierluigi Nuzzo. 2022. Towards a Formal Treatment of Logic Locking. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (Feb. 2022), 92–114. https://doi.org/10.46586/tches.v2022.i2.92-114
[10] Jitendra Bhandari, Abdul Khader Thalakkattu Moosa, Benjamin Tan, Christian Pilato, Ganesh Gore, Xifan Tang, Scott Temple, Pierre-Emmanuel Gaillardo, and Ramesh Karri. 2021. Not All Fabrics Are Created Equal: Exploring eFPGA Parameters For IP Redaction. https://doi.org/10.48550/arXiv.2111.04222 arXiv:2111.04222 [cs].
[11] Jitendra Bhandari, Abdul Khader Thalakkattu Moosa, Benjamin Tan, Christian Pilato, Ganesh Gore, Xifan Tang, Scott Temple, Pierre-Emmanuel Gaillardon, and Ramesh Karri. 2021. Exploring eFPGA-based Redaction for IP Protection. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. 1–9. https://doi.org/10.1109/ICCAD51958.2021.9643548 ISSN: 1558-2434.
[12] Abhishek Chakraborty, Nithyashankari Gummidipoondi Jayasankaran, Yuntao Liu, Jeyavijayan Rajendran, Ozgur Sinanoglu, Ankur Srivastava, Yang Xie, Muhammad Yasin, and Michael Zuzak. 2019. Keynote: A Disquisition on Logic Locking. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2019), 1–1. https://doi.org/10.1109/TCAD.2019.2944586 Conference Name: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.
[13] Rajat Subhra Chakraborty and Swarup Bhunia. 2010. RTL Hardware IP Protection Using Key-Based Control and Data Flow Obfuscation. In *2010 23rd International Conference on VLSI Design*. 405–410. https://doi.org/10.1109/VLSI.Design.2010.54 ISSN: 2380-6923.
[14] Jianqi Chen, Monir Zaman, Yiorgos Makris, R. D. Shawn Blanton, Subhasish Mitra, and Benjamin Carrion Schafer. 2020. DECOY: DEflection-Driven HLS-Based Computation Partitioning for Obfuscating Intellectual PropertY. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. 1–6. https://doi.org/10.1109/DAC18072.2020.9218519 ISSN: 0738-100X.
[15] Prattay Chowdhury, Chaitali Sathe, and Benjamin Carrion Schaefer. 2022. Predictive Model Attack for Embedded FPGA Logic Locking. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED '22)*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/3531437.3539728
[16] Alan R. Honorable Shaffer. 2021. A Microelectronic "Canary in a Coal Mine". (2021), 12. https://www.potomacinstitute.org/steps/featured-articles/96-a-microelectronic-canary-in-a-coal-mine

[17] Bo Hu, Jingxiang Tian, Mustafa Shihab, Gaurav Rajavendra Reddy, William Swartz, Yiorgos Makris, Benjamin Carrion Schaefer, and Carl Sechen. 2019. Functional Obfuscation of Hardware Accelerators through Selective Partial Design Extraction onto an Embedded FPGA. In *Proceedings of the 2019 on Great Lakes Symposium on VLSI (GLSVLSI '19)*. Association for Computing Machinery, New York, NY, USA, 171–176. https://doi.org/10.1145/3299874.3317992

[18] Yasaswy Kasarabada, Suyuan Chen, and Ranga Vemuri. 2019. On SAT-Based Attacks On Encrypted Sequential Logic Circuits. In *20th International Symposium on Quality Electronic Design (ISQED)*. 204–211. https://doi.org/10.1109/ISQED.2019.8697421 ISSN: 1948-3287.

[19] Dirk Koch, Nguyen Dao, Bea Healy, Jing Yu, and Andrew Attwood. 2021. FABulous: An Embedded FPGA Framework. In *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '21)*. Association for Computing Machinery, New York, NY, USA, 45–56. https://doi.org/10.1145/3431920.3439302

[20] Nimisha Limaye, Animesh B. Chowdhury, Christian Pilato, Mohammed T. M. Nabeel, Ozgur Sinanoglu, Siddharth Garg, and Ramesh Karri. 2021. Fortifying RTL Locking Against Oracle-Less (Untrusted Foundry) and Oracle-Guided Attacks. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. 91–96. https://doi.org/10.1109/DAC18074.2021.9586314

[21] Nimisha Limaye, Emmanouil Kalligeros, Nikolaos Karousos, Irene G. Karybali, and Ozgur Sinanoglu. 2021. Thwarting All Logic Locking Attacks: Dishonest Oracle With Truly Random Logic Locking. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 40, 9 (Sept. 2021), 1740–1753. https://doi.org/10.1109/TCAD.2020.3029133 Conference Name: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

[22] Prashanth Mohan, Oguz Atli, Joseph Sweeney, Onur Kibar, Larry Pileggi, and Ken Mai. 2021. Hardware Redaction via Designer-Directed Fine-Grained eFPGA Insertion. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 1186–1191. https://doi.org/10.23919/DATE51398.2021.9473910 ISSN: 1558-1101.

[23] Md Rafid Muttaki, Roshanak Mohammadivojdan, Mark Tehranipoor, and Farimah Farahmandi. 2021. HLock: Locking IPs at the High-Level Language. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. 79–84. https://doi.org/10.1109/DAC18074.2021.9586159 ISSN: 0738-100X.

[24] NYU. 2021. CSAW LLC 2021. https://sites.google.com/nyu.edu/csaw-llc-2021

[25] Christian Pilato, Animesh Basak Chowdhury, Donatella Sciuto, Siddharth Garg, and Ramesh Karri. 2021. ASSURE: RTL Locking Against an Untrusted Foundry. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 29, 7 (July 2021), 1306–1318. https://doi.org/10.1109/TVLSI.2021.3074004 Conference Name: IEEE Transactions on Very Large Scale Integration (VLSI) Systems.

[26] Christian Pilato, Fabrizio Ferrandi, and Donatella Sciuto. 2011. A design methodology to implement memory accesses in High-Level Synthesis. In *2011 Proceedings of the Ninth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*. 49–58. https://doi.org/10.1145/2039370.2039381

[27] Christian Pilato, Francesco Regazzoni, Ramesh Karri, and Siddharth Garg. 2018. TAO: techniques for algorithm-level obfuscation during high-level synthesis. In *Proceedings of the 55th Annual Design Automation Conference (DAC '18)*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/3195970.3196126

[28] Masoud Rostami, Farinaz Koushanfar, and Ramesh Karri. 2014. A Primer on Hardware Security: Models, Methods, and Metrics. *Proc. IEEE* 102, 8 (Aug. 2014), 1283–1295. https://doi.org/10.1109/JPROC.2014.2335155

[29] Jarrod A. Roy, Farinaz Koushanfar, and Igor L. Markov. 2010. Ending Piracy of Integrated Circuits. *Computer* 43, 10 (Oct. 2010), 30–38. https://doi.org/10.1109/MC.2010.284 Conference Name: Computer.

[30] Kaveh Shamsi, Meng Li, Kenneth Plaks, Saverio Fazzari, David Z. Pan, and Yier Jin. 2019. IP Protection and Supply Chain Security through Logic Obfuscation: A Systematic Overview. *ACM Transactions on Design Automation of Electronic Systems* 24, 6 (Sept. 2019), 65:1–65:36. https://doi.org/10.1145/3342099

[31] Kaveh Shamsi, David Z. Pan, and Yier Jin. 2019. IcySAT: Improved SAT-based Attacks on Cyclic Locked Circuits. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–7. https://doi.org/10.1109/ICCAD45719.2019.8942049 ISSN: 1558-2434.

[32] Yuanqi Shen, You Li, Amin Rezaei, Shuyu Kong, David Dlott, and Hai Zhou. 2019. BeSAT: behavioral SAT-based attack on cyclic logic encryption. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference (ASPDAC '19)*. Association for Computing Machinery, New York, NY, USA, 657–662. https://doi.org/10.1145/3287624.3287670

[33] Mustafa M. Shihab, Jingxiang Tian, Gaurav Rajavendra Reddy, Bo Hu, William Swartz, Benjamin Carrion Schaefer, Carl Sechen, and Yiorgos Makris. 2019. Design Obfuscation through Selective Post-Fabrication Transistor-Level Programming. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 528–533. https://doi.org/10.23919/DATE.2019.8714856 ISSN: 1558-1101.

[34] Dominik Sisejkovic, Luca Collini, Benjamin Tan, Christian Pilato, Ramesh Karri, and Rainer Leupers. 2022. Designing ML-Resilient Locking at Register-Transfer Level. In *Proceedings of the 59th Annual Design Automation Conference*.

[35] Dominik Sisejkovic, Farhad Merchant, Lennart M. Reimann, Harshit Srivastava, Ahmed Hallawa, and Rainer Leupers. 2021. Challenging the Security of Logic Locking Schemes in the Era of Deep Learning: A Neuroevolutionary Approach. *J. Emerg. Technol. Comput. Syst.* 17, 3, Article 30 (may 2021), 26 pages. https://doi.org/10.1145/3431389

[36] Pramod Subramanyan, Sayak Ray, and Sharad Malik. 2015. Evaluating the security of logic encryption algorithms. In *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. 137–143. https://doi.org/10.1109/HST.2015.7140252

[37] Xifan Tang, Edouard Giacomin, Aurélien Alacchi, Baudouin Chauviere, and Pierre-Emmanuel Gaillardon. 2019. OpenFPGA: An Opensource Framework Enabling Rapid Prototyping of Customizable FPGAs. In *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*. 367–374. https://doi.org/10.1109/FPL.2019.00065 ISSN: 1946-1488.

[38] Chiara Muscari Tomajoli, Luca Collini, Jitendra Bhandari, Abdul Khader Thalakkattu Moosa, Benjamin Tan, Xifan Tang, Pierre-Emmanuel Gaillardon, Ramesh Karri, and Christian Pilato. 2022. ALICE: An Automatic Design Flow for eFPGA Redaction. https://doi.org/10.1145/3489517.3530543 arXiv:2205.07425 [cs].

[39] Muhammad Yasin, Abhrajit Sengupta, Mohammed Thari Nabeel, Mohammed Ashraf, Jeyavijayan (JV) Rajendran, and Ozgur Sinanoglu. 2017. Provably-Secure Logic Locking: From Theory To Practice. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. Association for Computing Machinery, New York, NY, USA, 1601–1618. https://doi.org/10.1145/3133956.3133985

[40] Muhammad Yasin, Chongzhi Zhao, and Jeyavijayan JV Rajendran. 2019. SFLL-HLS: Stripped-Functionality Logic Locking Meets High-Level Synthesis. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–4. https://doi.org/10.1109/ICCAD45719.2019.8942150 ISSN: 1558-2434.

[41] Hai Zhou, Ruifeng Jiang, and Shuyu Kong. 2017. CycSAT: SAT-based attack on cyclic logic encryptions. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 49–56. https://doi.org/10.1109/ICCAD.2017.8203759 ISSN: 1558-2434.