You Break I Fix: A Collaborative Approach for Strengthening Sequential Obfuscation of Hardware Intellectual Property

Md Moshiur Rahman*, Travis Meade†, Yier Jin*, Swarup Bhunia*
*Department of Electrical and Computer Engineering
University of Florida, Gainesville, Florida 32611
Email: rahman.m@ufl.edu, yier.jin@ece.ufl.edu, swarup@ece.ufl.edu
†Department of Computer Science
University of Central Florida, Orlando, Florida 32816
Email: travis.meade@ucf.edu

***Abstract*—Hardware *intellectual property* (IP) infringement is a serious concern due to the horizontal model of the semiconductor supply chain. Several untrusted entities in this globalized supply chain are entitled to white-box accessibility of the reusable and valuable hardware IPs, making them vulnerable to piracy, cloning, tampering, and reverse engineering at different phases of system-on-chip (SoC) design and fabrication. Hardware obfuscation is a design transformation mechanism to protect the IPs from being exposed to the untrusted entities by locking the functionality of the IP while hiding the design intent. Unfortunately, existing obfuscation approaches are not resistant to various functional and structural attacks that are capable of unlocking the obfuscated designs. Due to the inability to analyze an obfuscation technique from the attackers' standpoint, numerous vulnerabilities inherent to the obfuscation methods go undetected unless a true adversary discovers them. In this paper, we present a collaborative approach between two entities - one acting as a *red team* and another as a *blue team* to replicate the real attacker-defender scenario, which in return strengthens the sequential obfuscation techniques. The *blue team* performs sequential obfuscation of gate-level IPs. On the other hand, the *red team* plays the role of an adversary/evaluator and tries to unlock the design by extracting the unlocking key or recovering the obfuscation circuitries.**

***Keywords*—FSM obfuscation; Reverse engineering; Red team-blue team**

I. Introduction

Hardware IP piracy, cloning, tampering, and reverse engineering is one of the most discussed problems related to the semiconductor life cycle that arise due to some common vulnerabilities in the state-of-the-art behavioral design of the IPs. Almost all real IPs have control logic to guide the operation of a circuit, and a common form of the control logic is a finite state machine (FSM). In a gate-level abstraction of the hardware IP, an FSM is a group of registers or flip-flops (FFs), referred to as *state flip-flops*, that controls the operation of a circuit by traversing through a number of states where the number of flip-flops in the state machine defines the number of states or state-space of that FSM. In the real circuit design scenarios, the size of the FSM is quite small, and the ratio of the state-space being used to the state-space available, or in another term, the reachability is quite low [1]. These deficiencies make the FSMs vulnerable to reverse engineering. An adversary can take advantage of this small size and easily reverse engineer the smaller state-space of the FSM to get access to the control logic part of the IP with minimal effort [2]. In contrast, a larger state-space of an FSM makes it harder for an adversary to extract the state-space.

Sequential obfuscation is a technique that alleviates the most significant vulnerability inherent to the IP design. A significant amount of effort has been put together to address these vulnerabilities in the sequential IP design. HARPOON [3], dynamic state deflection [4], active hardware metering [5], and some other techniques have been proposed to counter the IP invasion problem. However, the design community is still looking for a comprehensive solution to the specific problem as most of the evolved techniques are not resistant to emerging attacks. It is a necessity to create a reliable obfuscation mechanism as well as validate the strength through an appropriate and in-field approach. Fig. 1 replicates a robust collaborative approach of an attack-defense series going back and forth between two teams - the *red team* and the *blue team*. As a solution to the vulnerabilities associated with less

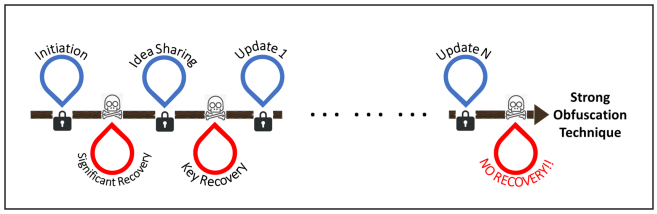


Fig. 1. A series of break (by *red team*) and fix (by *blue team*) to mitigate the vulnerabilities and strengthen sequential obfuscation.

reachable state-space, a *blue team* uses the flow of Fig. 2 which prevents the enumeration of state space and protects IP rights. The robustness of protection depends on the efficiency of the resistance to all known functional and structural attacks that apply to sequential obfuscation. To validate the efficiency, the *red team* applies a known set of attacks on the obfuscated designs to break it. Both the *red team* and the *blue team* goes through a series of exhaustive attacks and defenses to get a secure IP obfuscation technique. This series continues until the *blue team* succeeds in preventing the *red team* from being able to recover the secrets of the obfuscated design.

The rest of the paper is organized as follows. In Section II, we discuss different features of the proposed obfuscation methodology of *blue team*. In Section III and IV, we present the evaluation of the obfuscation technique by providing the details of the deobfuscation methodology of *red team*. In Section V, we describe the incremental changes that have been made by *blue team* to address the vulnerabilities found by deobfuscation methodology. We summarize the outcomes of the attack-defense approach in Section VI and conclude in Section VII.

II. Obfuscation Methodology

Sequential obfuscation hides the design intent by locking the functionality and changing the structure of a sequential design. The obfuscation mechanism relies on obfuscating the original FSM and inserting new FSM in addition to the obfuscated FSM. Fig. 2 illustrates the methodology of obfuscating the existing FSM and adding the extra FSM. The *blue team* uses the following high-level flow of sequential obfuscation to resolve the problems related to state-space enumeration.

A. State-Space Modifications

The state flip-flops in a design constitute the finite-state machine and are responsible for dictating the design through a finite number of states. For reachability analysis, the *blue team* executes state flip-flop identification algorithm to find a suitable set of flip-flops. Once identified, an artificial FSM (*NLFSR*) superimposition is performed to transform the functionality as well as the structure of the design FSM. Due to the superimposition of *NLFSR*, the original FSM is now configured to run into either of the obfuscated mode (*NLFSR*) or normal mode (original FSM). If the design is configured to run in obfuscated mode, the original FSM acts as an *NLFSR* which has 100% reachability; in other words, the FSM is reachable to all possible states. Superimposition is performed by adding multiplexers at the input of the target flip-flops as shown in Fig. 3a. The superimposition through multiplexing interprets the functional and structural transformation of the original FSM as well as the design.

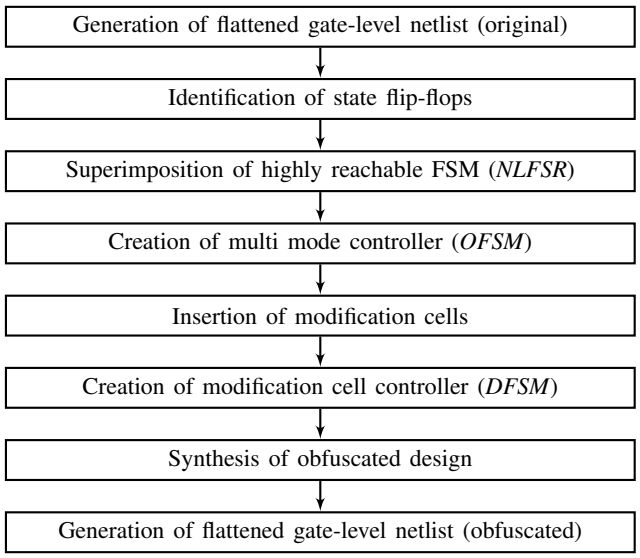


Fig. 2. Sequential obfuscation flow: original netlist to obfuscated netlist.

B. Multi Mode Controller

As discussed in the previous section, the obfuscated design can operate in two different modes. A configuration block is required to control the mode of operation. The configuration block is another FSM called *obfuscation FSM* (*OFSM*). *OFSM* is a simple finite-state machine that traverses through a set of states provided that a specific sequence of inputs is applied. This specific sequence is called *key sequence* which is the secret of the obfuscated design. Only the designer has the knowledge of this key sequence. After applying this specific key sequence through the existing primary inputs, the *OFSM* can be brought to an *enable state*, where the original FSM is enabled by selecting the appropriate input branch of the added multiplexers. In all other scenarios, obfuscated mode is enabled. The enable signals are generated by the *OFSM* and act as the select signal for the multiplexers. The gate-level implementation of the multiplexing of *NLFSR* and original design using *OFSM* is shown in Fig. 3b. Based on the size and other constraints of the design, more than one *OFSM* can be added and enable signals can also be a vector rather than one-bit signal which increases the difficulty of enabling the design without the knowledge of the key sequence. Due to the superimposition, the state-transition diagram adapts both obfuscated and normal modes as replicated in Fig. 3c for the obfuscated version of a small sample design. The state-space of the obfuscated design is larger compared to the original FSM. Hence, prevention of state enumeration, when running in the obfuscated mode, can be prevented for a significantly large number of flip-flops present in hardware IPs.

C. Data-Path Obfuscation

In addition to the control-path (FSM) obfuscation, *blue team* performs combinational data-path obfuscation with the help of insertion of modification cells. Modification cells can be fundamental key gates, e.g., XOR/MUX/AND/OR, or a custom boolean logic function which can be configured to be a buffer provided that correct enabling key values are applied to these cells. Insertion of modification cells can be done in a way that the output corruptibility, as well as the design structure, is changed to an extent from where it is hard to

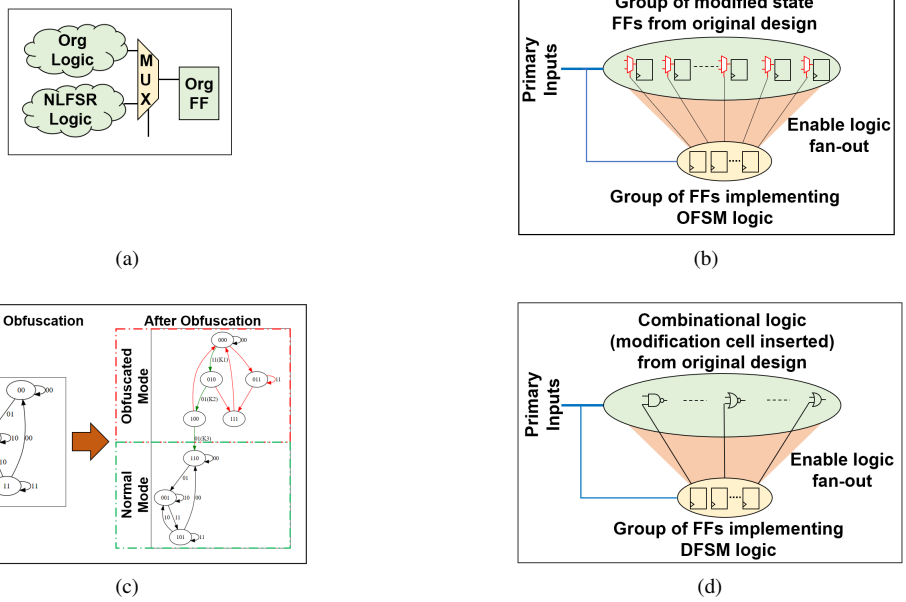


Fig. 3. State-space obfuscation: (a) multiplexing of superimposed FSM and original FSM (b) top-level diagram of obfuscated FSM controlled by added OFSM; (c) state transition diagram of the original design before and after obfuscation; (d) modification cell insertion.

reconstruct the original design or the enabling key values. Fig. 3d illustrates the gate-level insertion of the modification cells to obfuscate the combinational logic of a design. The unique feature of this combinational locking that differs from the existing logic locking techniques [6] is that to provide the enabling keys, *blue team* do not use additional primary inputs. Instead of adding visible key inputs, another controller, called *dummy FSM* (*DFSM*), is added to the design which generates the enabling key values. The structure and functionality of the *DFSM* is similar to the *OFSM*. The advantage of adding dummy FSM is that a sequence of keys is now required to configure the *DFSM* to operate in the enable state and generate the correct enable values rather than feeding the fixed-length key value through the primary inputs which increases SAT attack [7] resistance as well.

III. Evaluation

State-of-the-art obfuscation techniques, in most cases, lack proper scrutiny before these are claimed to be the robust and unbreakable. Only a real adversary can play important role in scrutinizing and detecting the flaws in any new or evolving techniques. Above presented state-space obfuscation also required to go through a series of attacks and vulnerability assessment from the *red team*'s point of view to accomplish the goal of having an obfuscation technique that is resilient to all known attacks. To validate the robustness of the sequential obfuscation technique, the *blue team* obfuscates gate-level designs, and shares different sizes of the obfuscated designs and algorithm with the *red team*. The shared designs are the obfuscated version of different open-source benchmarks. The *red team* tries to perform the role of an in-field attacker to unlock the design and provides the details of recovery.

IV. Deobfusction Methodology

The *red team* set three goals when attacking designs received from the *blue team*: 1) identify minimal FSM logic, 2) recover the high-level FSM, and 3) identify functioning states. To avoid poisoning the results the *red team* avoided identifying and comparing the obfuscated designs against unlocked third party designs.

A. State Element Identification

Perhaps the most critical problem to design recovery is that of FSM's state element identification. With a bad selection of flip-flops a number of problems arise when extracting an FSM. The following four cases of bad flip-flop selection analyzed are listed from least problem to most detrimental,

- Some critical, but not all critical state elements are selected;
- All state elements are selected, but so are non-state elements;
- Only non-critical state elements are selected;
- No state elements are selected.

A critical state element is a state element whose value is contingent on the value of other state elements in the previous clock cycle. This makes identification of other non-identified state elements easier. When generating the FSM with conditional information on transitions, the *red team* can look into other flip-flops that appear to have a significant (up to the interpretation of the *red team*) affect on the transitions of the state machine and include them in the potential state elements for future testing purposes. This case of incorrect selection becomes a minor issue.

When state elements and non-state elements are selected the "recovered" state graph gets bloated. Using ROBDD solvers to generate conditional transitions grows resulting graphs exponentially with respect to the extra non-state element flip-flops since each one adds an extra depth in terms of both potential transitions and states. On smaller designs (around 8 registers) tools that non-affecting FSM can accurately separate these state machines. With even larger selections more aggressive methods for FSM recovery are needed. We can manually try pairs/triplets of registers and identify which designs have the most synergy (again up to the interpretation of the *red team*). These selected pairs form edges in a graph representing the relationship between different flip-flops, where each node is a different flip-flop. The *red team* used connected components to infer which registers belong together in logical FSMs.

Absence of critical state element selection is the first non-detectable problem the *red team* encountered. Mitigation approaches involved looking to the output flip-flops of the selected flip-flops and trying to merge the close flip-flops into more complex FSMs, but due to some potential state element sets producing extremely large FSMs getting the process automated was tricky and the *red team* performed most of such analysis by hand.

The last problem the *red team* ran into was when the identified flip-flops did not compose the the state elements. Without state elements recovery was extremely challenging. A lack of state element selection is not always obvious when, since even one register can appear to form some logical FSM when run through analysis tools. This case can cause an attacker to venture down an attack strategy that has no reward, which seems to be the best deterrent for attackers.

B. High-Level FSM Recovery

Once the correct state elements have been selected the *red team* was typically able to recover the correct design fairly quickly. At first a slightly modified ROBDD solver was used for extraction. The *red team* was able to use the solver for most of the early protection schemes. However, as the designs grew more complex the *red team* developed a new heuristics for FSM recovery. In some designs the ROBDD method extracted FSMs where each condition had many "extra" inputs affecting the output of the FSM. Some inputs, when examined manually, appeared to have no real effect on the high level design, aside from reducing the rate of FSM recovery. To silence these non-affecting input signals, the *red team* set most of these wires to logical 0. After doing so, many of the designs were recovered in less than a second.

With this improvement, there was no guarantee that the recovered FSM was in fact the correct one. The *red team* concluded that the *blue team* might have incorporated rare triggering transitions using these inputs to transition to the functioning FSM. The *red team* utilized state-of-the-art 3-SAT solvers to check if any omitted transitions could be used. The 3-SAT solver found no extra states or transitions that was not already found by the ROBDD method. This leaves some other protection schemes for future approaches.

C. Unlocking Sequence Recovery

The last part of the *red team* goal was to extract the input sequence used to unlock the design. The *red team* assumed that like many logical locking methods, once a functional state was reached, transition to a non-functioning (obfuscation) state becomes impossible. This FSM behavior allows attackers to assume that the states in the FSM that comprise the functional states will form a Strongly Connected Component (SCC) which have no outgoing edges to other SCCs. A problem arises if the full FSM contains not just functional states as terminal SCCs, but obfuscation states as well. The *red team* assumed for later designs that the functional states would have a lower probability being reached overall. The *red team* devised an approach that would simulate the device for long periods of time to determine the probability of a device residing in every possible states. However, the approach was found to be unnecessary, since the state machine had few terminal SCCs, typically only one of which was obviously difficult to reach.

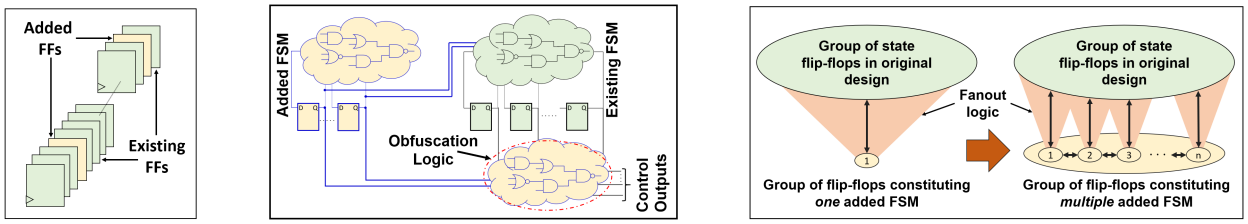
As a secondary validation strategy, the *red team* extracted each design's Finite State Transducer (FST), which models the finite state machine with the output created from each transition. The extracted FST gave insight to the correct functional states, as their output sequence remained constant for certain subsets of wires, while the obfuscation states had outputs that varied seemingly without reason. Lastly the *red team* used the extracted FST to build the RTL of the high level design to indicate what transitions/input sequences were necessary to unlock the design.

V. Revisiting The Vulnerabilities

From a series of attack-defense by both *red team-blue team*, it is evident that in sequential obfuscation, the added flip-flops are the critical assets of the obfuscated design and a point of interest for an attacker. The strength of the obfuscation depends on the efficiency of hiding the *OFSM*/*DFSM* flip-flops among all other flip-flops in the obfuscated design as shown in Fig. 4a. Efficient hiding makes it harder for an attacker to identify the added FSM flip-flops. As discussed in Section IV, the *red team* aims to find the added flip-flops from a large number of flip-flops in an obfuscated design and then use reverse engineering to reconstruct the *OFSM*/*DFSM*, recover the key sequence, or functional states. Based on the facts of the recovery, the *blue team* makes incremental changes to the implementation algorithm.

A. Control Output Dependency

The most significant vulnerability left off in the obfuscated design by the *blue team* is that the added FSMs do not affect the control outputs of the original design. In any hardware IP, there can be multiple control signals coming out from the control logic (FSM). Traversing the fan-in cones of these control signals reveal the control FSM of the design. The *blue team* failed to make these control signals dependent on both the original design FSM and the added FSM which makes it easier for the *red team* to isolate the added flip-flops from all others. This vulnerability is a viable point of attack and reveals



(a) (b) (c)

Fig. 4. Vulnerability assessment of FSM obfuscation : (a) *red team* aims to find the flip-flops added by the *blue team* from a sea of flip-flops; (b) *blue team* mitigates isolation of added flip-flops by changing fan-in cones of the control outputs; (c) *blue team* mitigates isolation of added flip-flops by breaking the fan-out logic distribution in multiple units of FSMs.

the added flip-flops as well as the unlocking key sequence. In response, the *blue team* addresses this serious security flaw by making the control outputs dependent on the added FSM as well. The gate-level implementation of this mechanism is illustrated in Fig. 4b. Dummy logic is added to the fan-in cones of the control outputs which is driven by the added FSMs. As a result, the control outputs are parts of both FSMs and an attacker may not be able to differentiate between added flip-flops and existing design flip-flops. As two different FSMs now drive the control outputs, both FSMs act as one combined FSM, making it difficult to reverse engineer one large FSM.

B. *Few$_{added}$-to-Many$_{existing}$ Tree of Flip-Flops*

In a large IP, it is possible to have hundreds of state flip-flops. In the initial version of the obfuscated design, the *blue team* added one new FSM containing a few flip-flops (5 to 8) to generate the enable signals that are responsible for driving hundreds of the multiplexers inserted into the obfuscated design. As a result, the *red team* can analyze the flop-to-flop mapping and discover that a small group of flip-flops is driving another large group of flip-flops which leak the information of the added flip-flops. To counter that problem, the *blue team* added multiple interconnected *OFSM*s, as shown in Fig. 4c, to distribute the fan-out of each added FSM. In addition to that, to break high fan-out logic towards a large number of modification cells from one FSM, multiple *DFSM*s are inserted as well. This mechanism has two major advantages, flop-to-flop mapping tree is broken into parts, and the *red team* have to deal with a large number of overlapping FSMs (which act as a one large FSM) to recover the individual FSMs.

VI. Red Team-Blue Team Practice Benefits

The unique *red team - blue team* approach presented here replicates the exact attacker-defender situation in integrated circuits (IC) design flow. Any new or evolving obfuscation technique should consider this approach which helps to increase level of confidence regarding the security of the proposed techniques. The benefits of this approach are summarized below.

- Evolution of a new sequential obfuscation technique;
- Evolution of a series of in-field attacks on sequential obfuscation technique;
- Opportunity to verify the claims in terms of security of the obfuscated design;
- An acknowledgement of both attacker and defender supporting the fact that the evolved sequential obfuscation is resistant to latest attacks;
- A provably secure and reliable hardware obfuscation technique.

VII. Conclusion

We have presented a netlist-level hardware obfuscation based anti-piracy design technique of sequential IPs which has the potential of being integrated into the design flow of modern SoC. Different levels of attacks have been performed to make sure that the obfuscation scheme does not leave any traces for an attacker to recover the design secrets through a red team-blue team effort. The presented scheme received scrutiny to such extent that it can be claimed to be robust enough to defeat any known attacks that apply to sequential obfuscations.

Acknowledgements

The work is supported by the Defense Advanced Research Projects Agency (DARPA).

References

[1] S. Kajihara, H. Shiba, and K. Kinoshita, "Removal of redundancy in logic circuits under classification of undetectable faults," in *[1992] Digest of Papers. FTCS-22: The Twenty-Second International Symposium on Fault-Tolerant Computing*. IEEE, 1992, pp. 263–270.

[2] T. Meade, S. Zhang, and Y. Jin, "Netlist reverse engineering for high-level functionality reconstruction," in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2016, pp. 655–660.

[3] R. S. Chakraborty and S. Bhunia, "Harpoon: an obfuscation-based soc design methodology for hardware protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 10, pp. 1493–1502, 2009.

[4] J. Dofe and Q. Yu, "Novel dynamic state-deflection method for gate-level design obfuscation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 2, pp. 273–285, 2017.

[5] Y. Alkabani and F. Koushanfar, "Active hardware metering for intellectual property protection and security." in *USENIX security symposium*, 2007, pp. 291–306.

[6] M. Yasin and O. Sinanoglu, "Evolution of logic locking," in *2017 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*. IEEE, 2017, pp. 1–6.

[7] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2015, pp. 137–143.