Fault Insertion based Attack on Logic Locking

M.Tech Project Stage 1 Report

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology

by

Nimesh Shedge

(Roll No. 20307R006)

Under the guidance of

Prof. Virendra Singh

Department of Electrical Engineering
Indian Institute of Technology Bombay
October 2022

Acknowledgement

I express my gratitude to my guide Prof. Virendra Singh for providing me the opportunity to work on this topic.

Nimesh Shedge
Electrical Engineering
IIT Bombay

Abstract

Logic Locking is used as a countermeasure against IP piracy and overproduction where key gates and inputs are added to the existing gate level netlist so that correct functionality is only obtained when the correct key is applied. SAT attack [1] was very effective against the encryption schemes then. Hence, Compound Logic Locking was adopted to resist the SAT attack. It consists of both high corruptibility and low corruptibility blocks. Fault Aided SAT attack [2] makes use of fault insertion to employ the original SAT again on Compound Logic Locking. But it needs functional verification. In this report, a method is explored for fault insertion such that the functionality remains the same. This will ensure that when SAT attack is applied, the correctness of the key is guaranteed.

Contents

List of Figures

Chapter 1

Introduction

There are various entities involved in the VLSI design flow because of the increasing cost of designing and manufacturing IC. Most of the ICs are outsourced to foundries for manufacturing. This can pose a number of challenging security threats such as IP piracy and IC overproduction.

To protect against these threats, various techniques have been introduced. The term logic locking was coined by EPIC [3]. It involved the introduction of key gates in the given gate-level netlist. Post manufacturing, the correct key value is loaded to restore the original functionality. The key inputs could be loaded from and stored in a tamper-proof on-chip memory. In this way, key gates hide the functionality of IC from untrusted entities. Even if the attacker can extract the gate-level netlist from reverse engineering, it will be locked because of the key gates. 'n' key inputs can have 2^n possibilities. Original functionality is only recovered if the correct key is loaded.

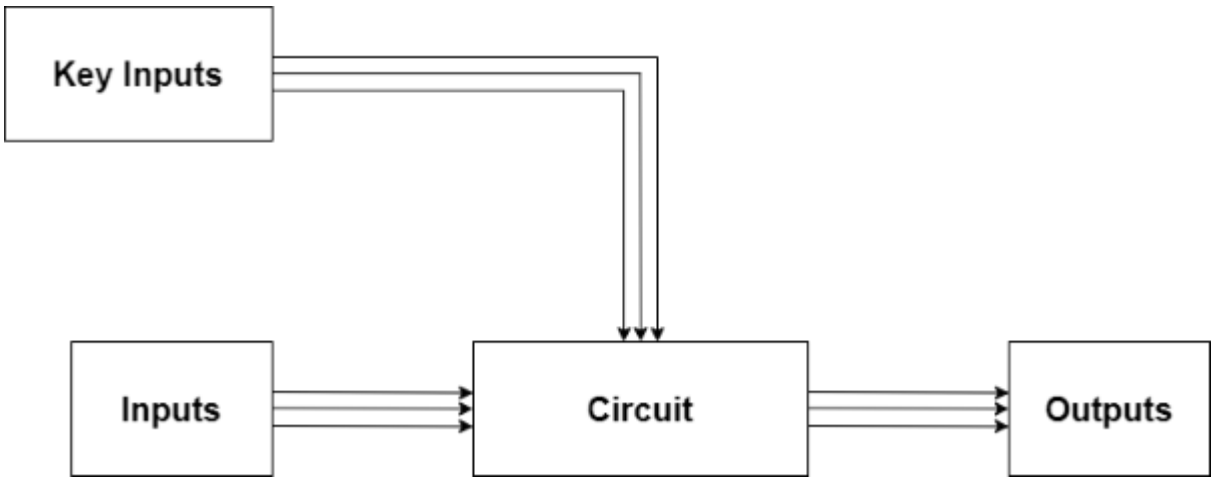

Figure 1.1: Overview of Logic encryption

Chapter 2

Literature Survey

Logic Locking is a strong defense against hardware security vulnerabilities like IP piracy and overproduction. EPIC [3] inserted key gates at random locations inside the gate-level netlist. Without the correct key, one will get incorrect functionality. But this was vulnerable to sensitization attack [4]. In this attack, stuck at fault is inserted in one of the key inputs, and a test pattern is generated using ATPG such that all other key inputs are don't care bits. The primary output to which the fault is propagated is also noted. Then this test pattern is applied to the working chip and the value at the said primary output is the correct key value of the corresponding key input. [4] also provided a defense against the attack. The key gates should be inserted such that there is maximum interference between two or more key inputs. This will ensure that one cannot generate a test pattern with all key inputs as don't care bits. Apart from this, [3] didn't necessarily corrupt or scramble outputs with incorrect keys. [5] made use of testing techniques to ensure that wrong keys caused wrong output values. The key gates are inserted such that the incorrect key effect is propagated to the primary output.

Prior to 2015, most encryption techniques focussed on increasing output corruption. In 2015, a powerful SAT attack [1] was introduced. It consisted of finding a distinguishing input pattern (DIP) such that two different key values produce different outputs for the same primary input value. Then this DIP is queried to the working chip (oracle) obtained from the market to find the correct output. This DIP along with its corresponding correct output is added as a clause for the SAT solver. Therefore, SAT attack tries to eliminate wrong keys in every iteration and is finally left with the correct key. The strength of the SAT attack lies in the number of keys it eliminates in a single iteration. [6] and [7] locks the netlist in such a way that only one key is eliminated per iteration. This is done with the help of a one-point function. Both encryption schemes differ in the way they implement the one-point function. [6] uses comparator while [7] uses complementary blocks such that one block is close to a one-point function for maximum sat resiliency. But these encryption scheme suffers from low output corruption, i.e. most of the output values are correct for incorrect keys. Also, they are implemented as separate blocks which makes them vulnerable to removal attacks [8]

Signal Probability Skew (SPS) attack [8] uses structural analysis to find the nets having maximum skew in signal probability. This is then used to identify the Anti SAT block and subsequently remove it. In addition to removal attack, SAT resilient schemes suffer from low output corruptibility. Bypass Attack [9] makes use of this fact and tries to correct the wrong key by adding bypass circuitry for its incorrect output. This attack has low area overhead only in case of low output corruptibility. Since SAT resilient schemes have low output corruptibility, they are combined with high output corruptibility schemes like Strong Logic Locking [5] which is then called Compound Logic Locking. Appsat attack [10] is an approximate SAT attack that tries to reduce the compound logic locking scheme to its SAT resilient part. It gives a key having low error rate. After a given number of iterations, it finds the error rate of the current key retrieved. If it falls below the error threshold, then it gives that as the approximate key. Bit-flipping attack [11] also targets compound logic locking where it tries to find the correct value of key contributing to high output corruptibility and uses bypass attack for the low output corruptibility block. Double DIP [12] specifically attacks compound logic locking schemes having SAR-Lock. It uses two DIPs instead of one to find the correct key value of the high output corruptibility block. For the low output corruptibility block, bypass attack is used.

In order to resist removal attacks, TTlock [13] was introduced. The original gate level netlist output is flipped for a particular input pattern. Then a restore circuitry is added that will correct the output for this input pattern. Thus, the input pattern acts as a key. The more general case of TTlock is Stripped Functionality Logic Locking (SFLL) [14], where outputs are flipped for all input patterns at a given hamming distance from the correct key. This has more output corruption. SFLL-fault [15] does the same thing as SFLL but differs in terms of implementation. Here fault-insertion is used to functionally strip the circuit. But it has a high implementation cost in terms of LUT. SFLL-rem [16] addresses that using Engineering Change Order. But [14] has structural vulnerabilities which is exploited by Fall attack [17]. Using structural analysis, it first finds the patterns for which output is flipped. From those patterns, it deduces the correct key. [17] also provides a key confirmation algorithm to find out the correct key from a subset of keys.

There are other attacks on compound logic locking as well. [18] tries to find the correct value of the key responsible for high output corruptibility. It makes use of two methods. One method involves output to multiple key inputs and the other involves key input to multiple outputs relationship. [2] inserts fault in the locked netlist to reduce its size and uses SAT attack again to retrieve the key in fewer iterations. But the key recovered is not guaranteed to be correct. Hence, it requires functional verification.

Chapter 3

Proposed Idea

3.1 Problem

[2] has one important problem that it cannot ensure the correctness of key. Hence, it relies on functional verification. This can again take a lot of time which was saved earlier. An example is constructed where an incorrect key is retrieved because the fault is inserted randomly. It is as follows.

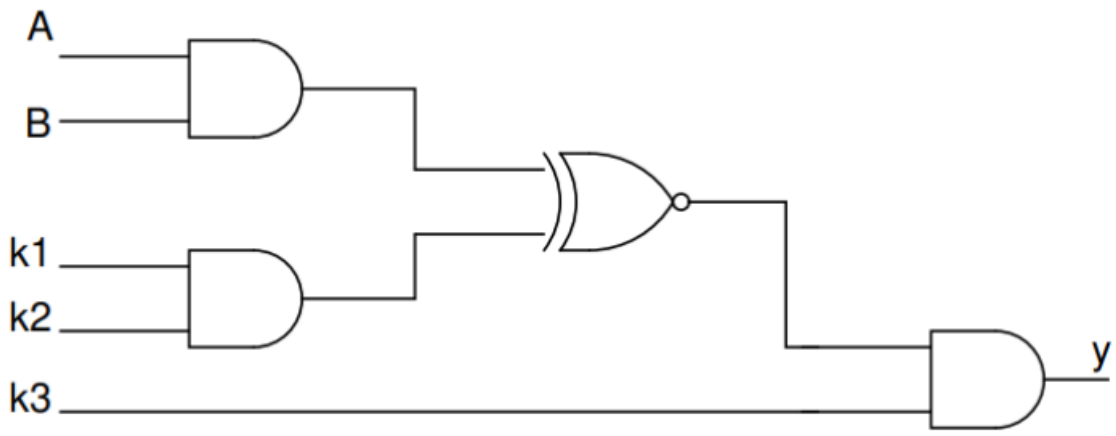

Figure 3.1: Original Locked Netlist

The key returned here is 000 (k1,k2,k3)

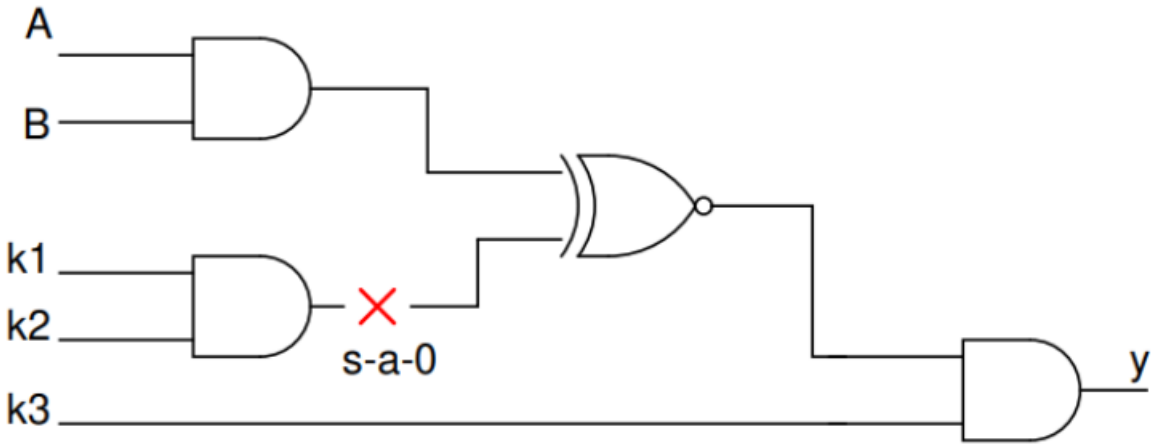

Figure 3.2: Fault Inserted Netlist

The key returned here can be 000, 010, 100, 110 (k1,k2,k3).

3.2 Proposed Solution

Incorrect key is returned because fault insertion flips the output for correct key with respect to previous circuit for all input patterns. To avoid that and to ensure the correctness of key, following flow can be adopted. The main idea is that the fault inserted netlist should be the same as the original locked netlist for some input patterns so that SAT attack can be applied without losing the correct key. The flow is as follows:

- After inserting the fault, find all the test patterns corresponding to it.
- Apply SAT attack on the fault inserted netlist without considering those test patterns as DIPs.
- When no DIPS are found, find all satisfying assignments to find the subset of keys.
- Use modified SAT attack on the original locked netlist to shortlist the key from the above subset

The problem with the above approach is that many test patterns exist for a particular fault at a given location.

3.3 Test Patterns as Distinguishing Input Patterns

In this method, a list of stuck at faults (0 or 1) at all primary inputs is made. For these faults, a set of test patterns is generated which can detect them. The generated test patterns are then used as Distinguishing Input Patterns (DIPs) in addition to all the other DIPs SAT solver will produce. This will aid the SAT solver in finding the correct key.

It was observed that the generated test patterns were not much effective in finding the key as compared to original DIPs. They did not eliminate more incorrect keys per iteration.

3.4 Future Work

Fault will be inserted in the primary inputs of locked gate level netlist. From this netlist, key will be inferred.

References

[1] 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)*, pp. 137–143, 2015.

[2] N. Limaye, S. Patnaik, and O. Sinanoglu, "Fa-sat: Fault-aided sat-based attack on compound logic locking techniques," in *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1166–1171, 2021.

[3] J. A. Roy, F. Koushanfar, and I. L. Markov, "Epic: Ending piracy of integrated circuits," in *2008 Design, Automation and Test in Europe*, pp. 1069–1074, 2008.

[4] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *Proceedings of the 49th Annual Design Automation Conference*, DAC '12, (New York, NY, USA), p. 83–89, Association for Computing Machinery, 2012.

[5] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault analysis-based logic encryption," *IEEE Transactions on Computers*, vol. 64, no. 2, pp. 410–424, 2015.

[6] M. Yasin, B. Mazumdar, J. J. V. Rajendran, and O. Sinanoglu, "Sarlock: Sat attack resistant logic locking," in *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 236–241, 2016.

[7] Y. Xie and A. Srivastava, "Anti-sat: Mitigating sat attack on logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 2, pp. 199–207, 2019.

[8] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Removal attacks on logic locking and camouflaging techniques," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 2, pp. 517–532, 2020.

[9] X. Xu, B. Shakya, M. M. Tehranipoor, and D. Forte, "Novel bypass attack and bdd-based tradeoff analysis against all known logic locking attacks." Cryptology ePrint Archive, Paper 2017/621, 2017. `https://eprint.iacr.org/2017/621`.

[10] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "Appsat: Approximately deobfuscating integrated circuits," *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 95–100, 2017.

[11] Y. Shen, A. Rezaei, and H. Zhou, "Sat-based bit-flipping attack on logic encryptions," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 629–632, 2018.

[12] Y. Shen and H. Zhou, "Double dip: Re-evaluating security of logic encryption algorithms," in *Proceedings of the on Great Lakes Symposium on VLSI 2017*, GLSVLSI '17, (New York, NY, USA), p. 179–184, Association for Computing Machinery, 2017.

[13] M. Yasin, A. Sengupta, B. C. Schafer, Y. Makris, O. Sinanoglu, and J. J. Rajendran, "What to lock? functional and parametric locking," in *Proceedings of the on Great Lakes Symposium on VLSI 2017*, GLSVLSI '17, (New York, NY, USA), p. 351–356, Association for Computing Machinery, 2017.

[14] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. J. Rajendran, and O. Sinanoglu, "Provably-secure logic locking: From theory to practice," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, (New York, NY, USA), p. 1601–1618, Association for Computing Machinery, 2017.

[15] A. Sengupta, M. Nabeel, M. Yasin, and O. Sinanoglu, "Atpg-based cost-effective, secure logic locking," in *2018 IEEE 36th VLSI Test Symposium (VTS)*, pp. 1–6, 2018.

[16] A. Sengupta, M. Nabeel, N. Limaye, M. Ashraf, and O. Sinanoglu, "Truly stripping functionality for logic locking: A fault-based perspective," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 12, pp. 4439–4452, 2020.

[17] D. Sirone and P. Subramanyan, "Functional analysis attacks on logic locking," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2514–2527, 2020.

[18] M. John, A. Hoda, R. Chouksey, and C. Karfa, "Sat based partial attack on compound logic locking," in *2020 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pp. 1–6, 2020.