THESIS TITLE

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

XX.

Contents

List of Figures

Chapter 1

Introduction

There are various entities involved in the VLSI design flow because of 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 [5]. It involved 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.

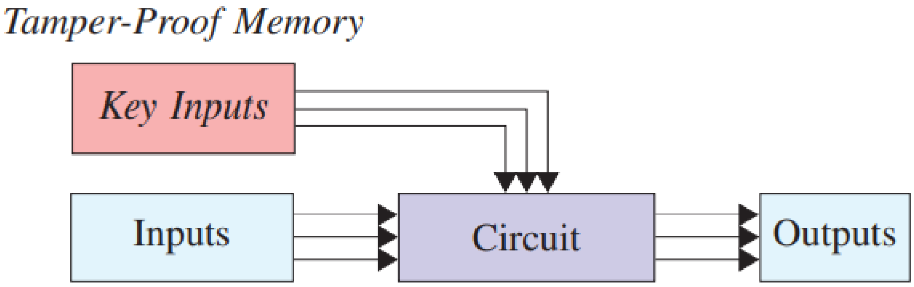

Figure 1.1: Overview of Logic encryption

Chapter 2

Literature Survey

2.1 SAT Attack [1]

Prior to 2015, most of the logic locking schemes focussed on increasing the output corruption of logic circuits. However in 2015, a potent attack was introduced, called SAT attack, which made use of boolean satisfiability to break most of the logic locking schemes. It requires two things to attack; one is locked gate level net-list and other being activated chip (oracle). In this attack, a miter circuit is constructed of the locked netlist such that there are two different keys but same primary inputs. The clauses are constructed for the above miter and a satisfying clause for primary input is found out such that outputs are different for two keys. This is referred to as distinguishing input pattern (DIP). DIP is then applied to oracle to find the correct output. The correct output for the given DIP is added as a constraint to SAT. It ensures that for the particular DIP, solver eliminates all the keys having outputs different from correct output. Thus, in one iteration multiple keys can be eliminated. This process is repeated until there are no DIPS. After that, the key which satisfies all the DIP constraints emerges as the correct one.

2.2 SAR lock [2] and Anti-SAT [3]

One of the measures that can be adopted to defend against SAT attack is to increase the number of SAT iterations. The encryption scheme should be such that the SAT attack can eliminate less number of keys per iteration, best being one. If only one key is eliminated per iteration, SAT will take exponential (2^{n}) no. of iterations to complete; rendering it ineffective. This can be done by one-point flipping function. SARlock and Anti-SAT both use this function to provide resistance against SAT. But this schemes suffer from low output corruptibility.

2.3 Fault-aided SAT-based Attack [4]

In this attack, a fault is inserted in the netlist randomly without performing any structural analysis. Then the SAT attack is performed on this fault inserted netlist to retrieve a key. For fault insertion at the proper node to result in a smaller circuit and, thus, fewer clauses and variables, a timeout of 10 and 90 seconds is offered. This allows for the return of the secret key in fewer iterations. If timeout is reached, fault is inserted at other node and the same process is repeated. Since the correctness of key is not guaranteed, functional verification is performed. It involves checking the key for some input patterns. In this way, key is extracted.

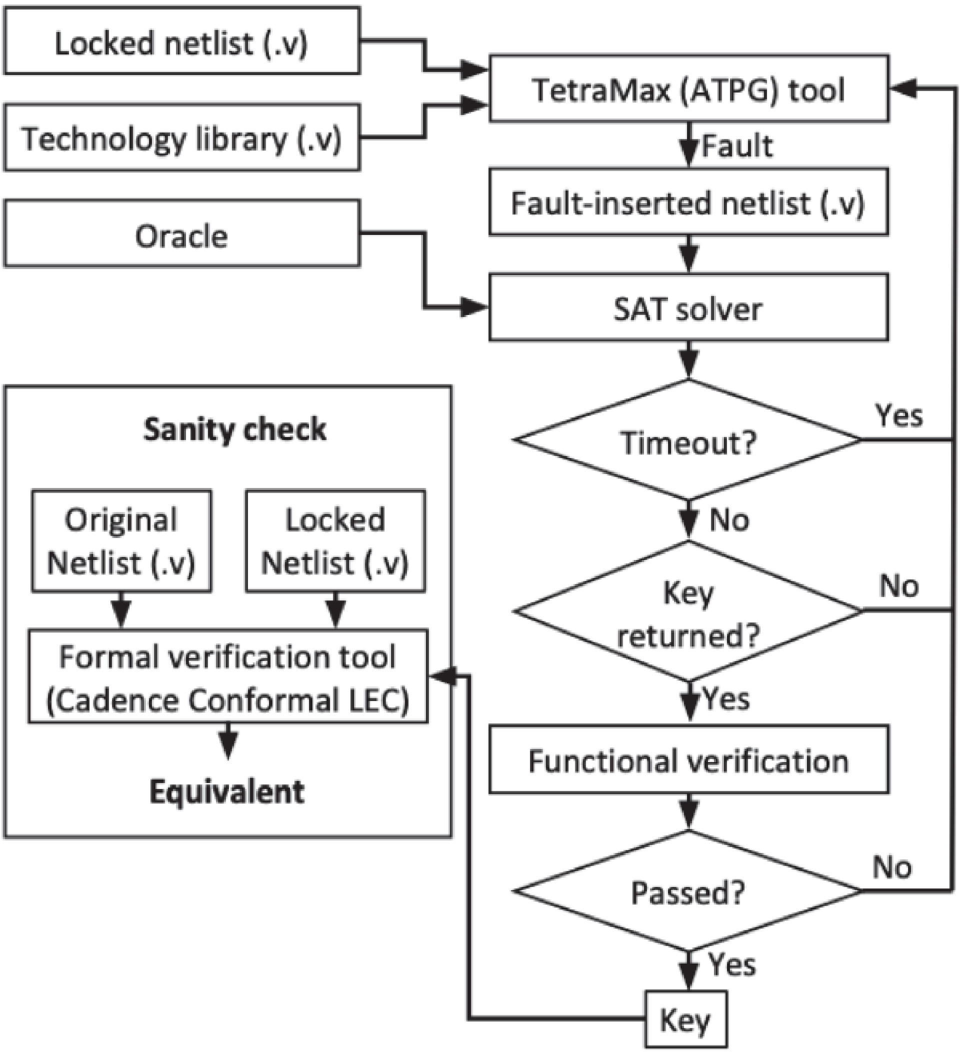

Figure 2.1: Fault-aided SAT-based Attack Flow

Chapter 3

Proposed Idea

3.1 Problem

[4] 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. I constructed an example where one can get an incorrect key because the fault is inserted randomly. It is as follows.

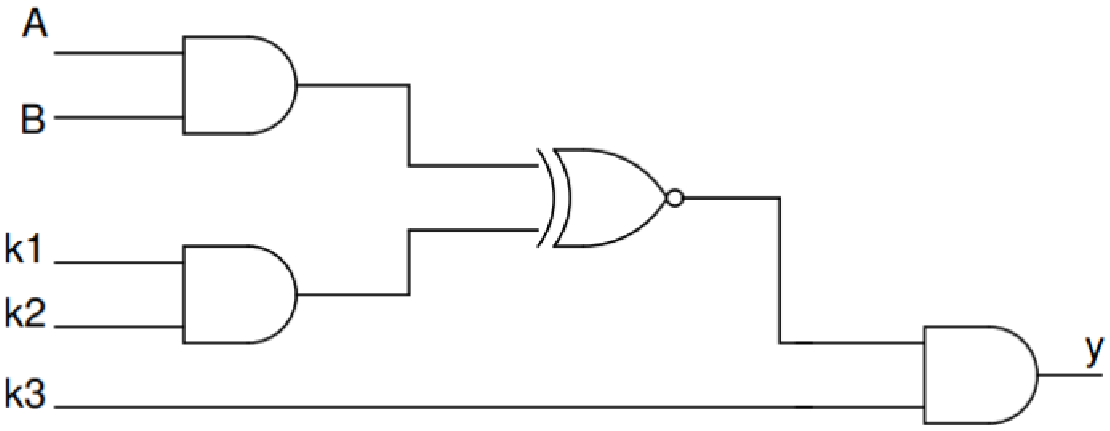

Figure 3.1: Original Locked Netlist

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

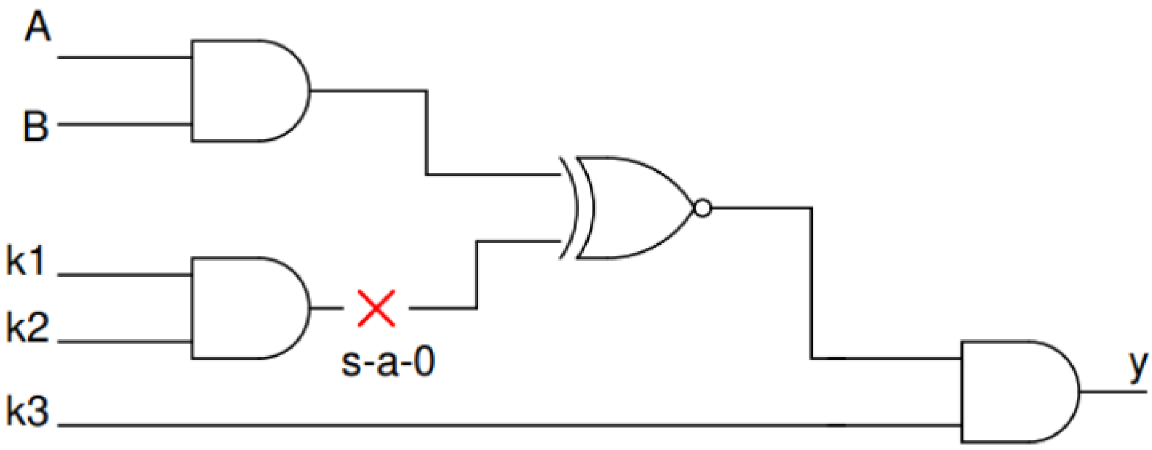

Figure 3.2: Fault Inserted Netlist

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 fault inserted netlist should be same as the original locked netlist for some input patterns so that we can apply SAT attack without losing the correct key. The flow is as follows:

- After inserting 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 above subset

The problem with the above approach is that many test patters exist for a particular fault at a given location. Hence, I am trying to find a way to circum-navigate it.

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] 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.

[3] 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.

[4] 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.

[5] 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.