Logic Locking a Design-for-Trust IC Design Technique
Marie-Lise Flottes

To cite this version:
Marie-Lise Flottes. Logic Locking a Design-for-Trust IC Design Technique. École d’hiver Francophone sur les Technologies de Conception des Systèmes Embarqués Hétérogènes (FETCH 2020), Feb 2020, Montréal, Canada. lirmm-03125660

HAL Id: lirmm-03125660
https://hal-lirmm.ccsd.cnrs.fr/lirmm-03125660
Submitted on 29 Jan 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
LOGIC LOCKING
A DESIGN-FOR-TRUST
IC DESIGN TECHNIQUE

M.-L. Flottes

LIRMM (CNRS - Université de Montpellier)
France

FETCH 2020
École d'hiver Francophone sur les Technologies de Conception
des Systèmes Embarqués Hétérogènes
COUNTERFEIT ICS: SOURCES & ISSUES

- Source: profit + globalization

  ![Diagram](https://via.placeholder.com/150)

- Issues: Financial loss/Reliability/Security
  - Miss out $100 billion/year
  - Reported counterfeit parts have been quadrupled since 2009
  - Many sectors are impacted (computers, telecom, automotive, .... military systems)
  - Dramatic consequences on critical systems
TAXONOMY

- Recycled/remarked components
  - Old components sold as new
  - New components sold with higher specification
    - commercial grade → industrial grade → defense grade
- Overproduction: Fabrication outside contract
  - Extra ICs or defective/out-of-spec components
- Cloning: Design copy
  - Reverse Engineering / IPs obtain illegally
- Tampered type: Hardware/Software Trojans (HT/ST)
  - Inserted at any level
  - Time bomb / back door
COUNTERFEIT DETECTION

- Physical detection
  - X-Ray, SEM
- Electrical detection
  - Parametric Tests / Functional tests

Time/Cost and Confidence issues
COUNTERFEIT AVOIDANCE

- “Need for development of innovative avoidance mechanisms to be incorporated in the design”
- (e.g. RO-Based) Sensors: Prevent die and IC recycling [15-16]
- Split manufacturing: Prevent overproduction [17]
- IC camouflaging: Prevent reverse engineering [18]
- Hardware watermarking: Secure IPs [19]
- Hardware metering:
  - Passive methods
    - Digitally stored serial numbers (nonfunctional identification)
    - PUF (functional identification)
  - Active methods: lock each IC until key is provided by the IP holder
    - Initialize IC to a locked state on power up
    - Add an FSM to unlock with the correct sequence to Initial State
    - Logic locking
OUTLINE

- Principle
- Implementations
- SAT Attack on logic locking
- Improvement on logic locking solutions and other attacks
- Conclusions
TECHNICAL PRINCIPLE: KEYING MECHANISM

Original Circuit

Protected Circuit

Taper-proof memory

Key inputs

Correct / Incorrect

PIs

POs
TECHNICAL PRINCIPLE: KEY GATES & KEY BITS

Original Circuit

**XOR Key gate**

<table>
<thead>
<tr>
<th>Key bit K1</th>
<th>K1=0</th>
<th>K1=1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>XNOR Key gate</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Key bit K1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K1=1</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>K1=0</td>
<td>✗</td>
<td>✔</td>
</tr>
</tbody>
</table>

**not(gate)**

<table>
<thead>
<tr>
<th>Key bit K1</th>
<th>K1=1</th>
<th>K1=0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>XNOR Key gate</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Key bit K1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K1=0</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>K1=1</td>
<td>✗</td>
<td>✔</td>
</tr>
</tbody>
</table>
EVALUATION

- Output corruptibility
  - HD(corret outputs, incorrect outputs)
  - Optimum HD = 50% (maximal ambiguity)

- Security
  - Possibilities to penetrate the system using techniques available to an attacker
APPLICATION PRINCIPLE IN THE IC DESIGN FLOW

TRUSTED DESIGN

- System Specs.
- Logic synthesis
- Netlist
- Logic locking
- Locked Netlist
- Physical synthesis

UNTRUSTED DESIGN/FABRICATION

- Locked chip
- Test & Packaging
- Locked IC
- Masks & Fab.
- Layout

TRUSTED KEY TRANSMISSION

- Activation (K from IP owner)

MARKET

- Functional Chip

✓ Prevents from Reverse Engineering
✓ Prevents from Overproduction
✓ Makes harder identification of ‘safe place’ for HT insertion
Acker knows the locked netlist / has un unlocked circuit (K inside)
OUTLINE

- Principle
- Implementations
- SAT Attack on logic locking
- Improvement on logic locking solution sand other attacks
- Conclusions
IMPLEMENTATION(S)

- First 2010
    - RLL: Random Logic locking
      - Introduce k XOR/NXOR key-gates at random locations (while meeting timing constraints)
    - LUT-based locking (Correct/incorrect LUT programming provide modification of the information flow)
      - Introduce LUT at choosen location for maximum attacker effort (low-controllable nodes), and for optimal output corruption (high observable nodes)
First improvements (output corruption)

- [8] 2015 « Fault Analysis-Based Logic Encryption »
  - **FLL: Fault-Analysis-based logic locking**
  - Introduce k XOR/NXOR key-gates at chosen locations for optimal output corruption
  - Metric (maximal number of patterns NC to control the node & maximal number of affected primary outputs NO)
  - Highest FI = NC₀xNO₀ + NC₁xNO₁
- [9] Variante 2017
  - **WLL: Weighted logic locking**
  - XOR key-gates fed by multiple key-bits through additional AND/OR gates which leads to a higher output corruptibility
First improvements (security)

**Issue**

Input Patterns $(e_1, e_2, e_3, e_4) = (1, 0, 1, x)$

S = K1 !

**SLL: Strong Logic Locking**

Introduce XOR/NXOR key-gates at chosen locations for ensuring interdependence among key bits

$S = K1^* \oplus K2^*$
OUTLINE

- Principle
- Implementations
- SAT Attack on logic locking
- Improvement on logic locking solutions and other attacks
- Conclusions
SAT ATTACK

  1/ Found a DIP (Differential Input Pattern) / $f(DIP,K1) \neq f(DIP,K2)$
  2/ Compare $f(DIP,Ki)$ with Oracle(DIP)
    - If $f(DIP,Ki) \neq$ Oracle(DIP), $Ki$ can be rejected
  3/ Iterate until no more DIP is found
    - All incorrect keys have been rejected

95% of experimented circuits are decrypted 90% with < 250 DIPs

Thwart all previous presented locking techniques

Netlist Copy - 1
Netlist Copy - 2
$S_1 = 1$
$S_2 = 1$
$K_2 = 1$
$K_1 = 0$
$DIP = 1$
OUTLINE

- Principle
- Implementations
- SAT Attack on logic locking
- Improvement on logic locking solutions and other attacks
- Conclusions
POST-SAT-ATTACK SOLUTIONS

- Resisting the SAT-attack by increasing its Execution time

\[ \text{SAT Execution Time: } ET = \sum_{i=1}^{\text{iter}} t_i \]

- \( \rightarrow \) Controlling the distinguishing ability of DIPs
- \( \rightarrow \) Rule out at most one incorrect key per DIP

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Original O</th>
<th>O for ki</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1 I2 I3</td>
<td></td>
<td>K0 K1 K2 K3 K4 K5 K6 K7</td>
</tr>
<tr>
<td>0 0 0</td>
<td>0</td>
<td>1 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 1</td>
<td>1</td>
<td>1 0 1 1 1 1 1 1</td>
</tr>
<tr>
<td>0 1 0</td>
<td>1</td>
<td>1 1 0 1 1 1 1 1</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0</td>
<td>0 0 0 1 0 0 0 0</td>
</tr>
<tr>
<td>1 0 0</td>
<td>0</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>0</td>
<td>0 0 0 0 0 1 0 0</td>
</tr>
<tr>
<td>1 1 0</td>
<td>1</td>
<td>1 1 1 1 1 1 0 1</td>
</tr>
<tr>
<td>1 1 1</td>
<td>0</td>
<td>0 0 0 0 0 0 0 1</td>
</tr>
</tbody>
</table>
**POST-SAT-ATTACK SOLUTIONS (CONT’D)**

- SARLock [13], 2016 « SAT Attack Resilient logic locking »

  ![Diagram](image)

- Anti-SAT, [14], 2019 « Preventing SAT Attacks »
  - SAT Execution time / output corruptibility Trade-off

- Low output corruptibility!

  ![Image](image)
OTHER ATTACKS ON LOGIC LOCKING

- Removal attacks
  - remove locking mechanisms from the studied netlist
- Approximate attacks on compound logic locking techniques (e.g., SARLock+FLL)
  - returns an approximate key (only FLL key bits are extracted) limiting the low-corruptability constituent in the netlist (SARLock countermeasure)
- Power side-channel attacks
- Oracle-less attacks (e.g., redundancy identification)
**CONCLUSION**

- **Design for Trust (DfTr)**
  - Watermarking that embeds a designer’s signature into the design
  - Passive metering that enables tracking of individual ICs throughout their lifetime
  - Camouflaging that introduces look-alike structures at the layout-level
  - Split manufacturing that involves partial fabrication at two separate foundries
  - And…

- **Logic locking**
  - Locks a design with key-controlled protection logic
  - Protection anywhere in the supply chain
    - Rogue SoC integrator (IP reuse)
    - Untrusted foundry (overproduction, HT)
    - Untrusted test facility (sell defective parts, recycling)
    - Malicious end-user (replicate)
• All logic Locking solutions exhibit specific weakness
• No metrics
• May exhibit vulnerabilities after implementation
• Implementation Cost
Merci !
REFERENCES

[0] http://www.blogpresidentcnac.fr/lutter-contre-la-contrefaçon-de-composants-electroniques/
