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

- 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

Foundry/package/test

EDA

IPs
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
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

PIs

POs

K_{correct} / K_{incorrect}

PIs

POs

Correct

Incorrect
TECHNICAL PRINCIPLE: KEY GATES & KEY BITS

Original Circuit

XOR Key gate

Key bit K1
K1=0 ✓ K1=1 ✗

XOR Key gate

Key bit K1
K1=1 ✓ K1=0 ✗

XOR Key gate

Key bit K1
K1=1 ✓ K1=0 ✗

XNOR Key gate

Key bit K1
K1=1 ✓ K1=0 ✗

XNOR Key gate

Key bit K1
K1=0 ✓ K1=1 ✗
EVALUATION

- Output corruptibility
  - HD(correct 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

- Prevents from Reverse Engineering
- Prevents from Overproduction
- Makes harder identification of ‘safe place’ for HT insertion
ASSUMPTION ON LOGIC LOCKING ATTACKS

- Acker knows the locked netlist / has un unlocked circuit (K inside)
OUTLINE

- Principle
- Implementations
- SAT Attack on logic locking
- Improvement on logic locking solution and 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**

- SLL: Strong Logic Locking
  - Introduce XOR/NXOR key-gates at chosen locations for ensuring interdependence among key bits

Input Patterns
(e₁, e₂, e₃, e₄) = (1, 0, 1, x)

S = K₁ !
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) ≠ f(DIP,K2)
  - 2/ Compare f(DIP,Ki) with Oracle(DIP)
    - If f(DIP,Ki) ≠ 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
K1
DIP
S1
K2
S2

Current = 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

  SAT Execution Time: \( ET = \sum_{i=1}^{iter} ti \)

- \( \Rightarrow \) 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>L1 L2 L3</td>
<td>K0 K1 K2 K3 K4 K5 K6 K7</td>
<td></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>1</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>

\( 2^k - 1 \) DIPs to succeed!
**POST-SAT-ATTACK SOLUTIONS (CONT’D)**

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

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

- Low output corruptibility !

- Removal Attacks !

I

logic cone

K
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) living 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-contrefacon-de-composants-electroniques/
