Protect Your Chip Design Intellectual Property: An Overview

Johann Knechtel, Satwik Patnaik, and Ozgur Sinanoglu
{johann, sp4012, ozgursin}@nyu.edu

COINS 2019, May 6, Crete, Greece
Threats for IC Fabrication and Hardware Security

Introduction

Logic Locking

Layout Camouflaging

Split Manufacturing

Summary

Kerry Bernstein, DARPA, 2016

Growing Demand for Protection of Design IP

**A Case Study in Fake Chips**

In 2010 the United States prosecuted its first case against a counterfeit-chip broker. The company, VisionTech, sold thousands of fake chips, many of which were destined for military products.

<table>
<thead>
<tr>
<th>Counterfeit parts sold by VisionTech</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motorola ICS</td>
</tr>
<tr>
<td>1500</td>
</tr>
<tr>
<td>Intel flash memory ICS</td>
</tr>
<tr>
<td>1500</td>
</tr>
<tr>
<td>Cypress ICs</td>
</tr>
<tr>
<td>350</td>
</tr>
<tr>
<td>Altera ICs</td>
</tr>
<tr>
<td>196</td>
</tr>
<tr>
<td>National Semiconductor ICs</td>
</tr>
<tr>
<td>75</td>
</tr>
</tbody>
</table>

Source: Sentencing memo, United States of America v. Stephanie A. McGlosh, filed 7 September 2011

**APRIL 2019: ZHENGZHOU CUSTOMS DESTROYS COUNTERFEIT TI CHIPS WORTH 704M YUAN**

Zhengzhou Customs seized 20,000 automotive CPU ICs labeled with the Texas Instruments (TI) trademark, suspecting them to be counterfeit. [...] The intended function of the CPUs was to prevent short circuits caused by instantaneous current overload when a vehicle is started. Total value of the fake chips was estimated at 704 million yuan. (around 100 million USD).

Protect Your Chip Design IP: An Overview

Basics of Logic Locking (Encryption)

- IP owner locks the design at RTL, by inserting dedicated locking structures
- IP owner unlocks the design after fabrication, by loading secret key onto memory
- Protects against untrusted end-user + fab
Basics of Logic Locking (Encryption)

- Incorrect key → Incorrect output

⚠️ Secure realization of tamper-proof memories
⚠️ Prone to analytical and invasive attacks

Evolution of Logic Locking

**Variants**
- RLL: DATE'08, Rice & Mich.
- FLL: DATE'12, NYU
- SLL: DAC'12, NYU

**Defenses**
- SARLock: HOST'16, NYU
- Anti-SAT: CHES'16, Maryland
- TTLock: GLSVLSI'17, NYU

**Algorithmic**
- Sens: DAC'12, NYU
- SAT: HOST'15, Princeton

**Removal**
- Bypass: CHES'17, UF

**Approx.**
- 2-DIP: GLSVLSI'17, NW
- AppSAT: HOST'17, UCF

**Side-channel Test-data mining De-synthesis**
- CycSAT: ICCAD'17, NW

**Attacks**
- SFL: CCS'17, NYU
- ATPG-based SFL: VTS'18, NYU

**Date**
- 2008
- 2012
- 2015
- 2016
- 2017

Boolean Satisfiability: A Powerful Attack on Logic Locking

Key search space

Iter 1
000 ... 01
Iter 2
000 ... 10
Iter 1
111 ... 11

Correct key

Locked netlist

(00011, 01)

SAT solver

DIP: Distinguishing input pattern

Functional IC

SAT attacks broke all basic logic locking techniques

### SAT Attack Success

<table>
<thead>
<tr>
<th>No.</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>Y</th>
<th>k0</th>
<th>k1</th>
<th>k2</th>
<th>k3</th>
<th>k4</th>
<th>k5</th>
<th>k6</th>
<th>k7</th>
<th>Pruned key values</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DIP 1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>✓</td>
<td>✓</td>
<td>Iter 1: {k4}</td>
</tr>
<tr>
<td></td>
<td>DIP 3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>✓</td>
<td>Iter 3: all incorrect</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DIP 2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>✓</td>
<td>Iter 2: {k1, k2}</td>
</tr>
</tbody>
</table>

**Attack success ≈ effectiveness and selection of DIPs**

**SAT Attack Success**

Worst-case scenario for attack: Each DIP can eliminate only one key

<table>
<thead>
<tr>
<th>No.</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>Y</th>
<th>k0</th>
<th>k1</th>
<th>k2</th>
<th>k3</th>
<th>k4</th>
<th>k5</th>
<th>k6</th>
<th>k7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>✓</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Worst case for attack:**

#DIPs = \(2^k - 1\)

**Trade-off:**

SAT attack resilience v/s output corruptibility

---


10/30
Point-Function-Based Logic Locking Techniques

- Integration of point functions
  - E.g., AND/OR tree
  - Allows to control error injected into circuit
- Renders number of DIPs exponential in key size
- **Vulnerability**: Structural traces (identify & remove)
Stripped Functionality Logic Locking (SFLL)

- Based on “strip and restore”
  - *Locked* circuit obtained from *original* circuit by making various changes at gate/RTL level
  - *Restore circuit* is intertwined

- In principle secure against all known attacks

- Quantifiable protection

<table>
<thead>
<tr>
<th>I2</th>
<th>I1</th>
<th>I0</th>
<th>original</th>
<th>locked</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Example: 3 protected input patterns; Error Rate = 3/8 (on locked output)
**SFLL Chip**

- First-of-its kind demonstration of resilient logic locking in 2017
- ARM Cortex-M0 microprocessor, 65nm GlobalFoundries technology
  - Layout cost affordable (1.6% A, 5.6% P, 5.4% D)
- https://github.com/DfX-NYUAD/CCS17
### Basics of Layout Camouflaging

- Alter the chip’s appearance to make it arduous for an attacker to infer the real functionality

#### Trade-offs for security and cost (manufacturing cost, layout cost)

#### Prone to invasive and also to analytical attacks

---

<table>
<thead>
<tr>
<th>Introduction</th>
<th>Logic Locking</th>
<th>Layout Camouflaging</th>
<th>Split Manufacturing</th>
<th>Summary</th>
</tr>
</thead>
</table>
Attacks on Layout Camouflaging

1) Modeling of unknown gates as locking problem, using SAT attacks
2) Etching, failure analysis, electron microscopy, photon emission, etc.

FEOL-Centric Layout Camouflaging

- Dummy contacts, e.g., NAND-NOR-XOR primitive in [Rajendran-CCS13]
  - PPA cost of 5.5X, 1.6X, 4X over 2-input NAND gate
  - Small-scale application, possibly locking-inspired; low error rate
  - Can be reverse-engineered using SEM PVC
FEOL-Centric Layout Camouflaging

- Threshold-dependent gates, e.g., NAND-NOR-XOR in [Akkaya-ISSCC18]
  - Post-manufacturing configurability, unlike static camouflaging
  - PPA cost of 9.2X, 6.6X, 7.3X over 2-input NAND gate
  - Doping can be reverse-engineered using SEM (PVC) or careful etching


Scanning Electron Microscopy Passive Voltage Contrast

BEOL-Centric Layout Camouflaging

- Dummy vias, wires in [Chen-DFTS15], [Malik-ISVLSI15], [Patnaik-ICCAD17]
- Simple to manufacture – only BEOL masks affected, any FEOL compatible
- No inherent gate-level cost
  - Full-chip camouflaging: SAT attack hindered by scalability issue
- BEOL materials: Mg/MgO vias in [Chen-DFTS15], [Patnaik-ICCAD17]
BEOL-Centric Layout Camouflaging

- Dummy vias, wires in [Chen-DFTS15], [Malik-ISVLSI15], [Patnaik-ICCAD17]
- Simple to manufacture – only BEOL masks affected, any FEOL compatible
- No inherent gate-level cost
  - Full-chip camouflaging: SAT attack hindered by scalability issue

- BEOL materials: Mg/MgO vias in [Chen-DFTS15], [Patnaik-ICCAD17]
  - Mg/MgO used in CMOS processes (for MTJs, Damascene process, ...)

---


---

BEOL-Centric Layout Camouflaging

- Dummy vias, wires in [Chen-DFTS15], [Malik-ISVLSI15], [Patnaik-ICCAD17]
- Simple to manufacture – only BEOL masks affected, any FEOL compatible
- No inherent gate-level cost
  - Full-chip camouflaging: SAT attack hindered by scalability issue
- BEOL materials: Mg/MgO vias in [Chen-DFTS15], [Patnaik-ICCAD17]
  - Mg/MgO used in CMOS processes (for MTJs, Damascene process, ...)
  - Difficult to reverse engineer: Mg oxidizes
  - Charge-based SEM may fail as well


Basics of Split Manufacturing

• Split the design process into multiple stages
  – Typically split into FEOL and BEOL
  – Good support of economics-driven supply chain

⚠ Trade-off for security and practicability (split layer, BEOL requirements, wafer handling)
⚠ Prone to analytical attacks

Attacks on Split Manufacturing – Proximity Attacks

- CAD tools work holistically on FEOL and BEOL

- Infer missing BEOL connections from FEOL layout [Rajendran-DATE13]
  - Placement proximity, direction of dangling wires

- Additional hints, various attack implementations
  - Load capacitance, no combinatorial loops, timing constraints [Wang-DAC16]
  - Routing proximity, estimated routing congestion [Magana-ICCAD16]
Defense Schemes

- Placement perturbation [Wang-DAC16]
  - Selective, small-scale use – proximity attack rate at 92%

- Routing perturbation [Wang-ASPDAC17], [Magana-ICCAD16], [Feng-ICCAD17], [Patnaik-ASPDAC18]
Defense Schemes

- Placement and routing perturbation – “netlist restructuring” [Sengupta-ICCAD17, Patnaik-DAC18]
  - Better security, proximity attack success rate as low as 0%
  - PPA for large-scale application


Patnaik et al.: Raise Your Game for Split Manufacturing: Restoring the True Functionality Through BEOL, Proc. DAC, 2018

Split Manufacturing for Protection Against Hardware Trojans

- When the fab attacker already knows the netlist, how to prevent Trojans? [Imeson13]  
  - Layout cost
- When the fab attacker inserted some Trojan, how to test for? [Vaidyanathan14]  
  - Commercial cost

Extending Split Manufacturing by 3D Integration

⚠ Prior art: high layout cost, commercial cost, protect only against fab

➡ “Best of both worlds”: split manufacturing and BEOL camouflaging

➡ Security-driven “3D split” into two (or more) tiers

Patnaik et al., Best of Both Worlds: Integration of Split Manufacturing and Camouflaging into a Security-Driven CAD Flow for 3D ICs
Proc. IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD), 2018, 8:1-8:8
Extending Split Manufacturing by 3D Integration

- Split manufacturing and BEOL camouflaging
  - Security-driven “3D split” into two (or more) tiers
  - Randomize and camouflage interconnects (RDLs)
  - Only trusted BEOL facility is required
  - Thwarts both malicious FEOL fabs and end-user

Patnaik et al., Best of Both Worlds: Integration of Split Manufacturing and Camouflaging into a Security-Driven CAD Flow for 3D ICs
Proc. IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD), 2018, 8:1-8:8

Protect Your Chip Design Intellectual Property: An Overview

- Complex and globalized, outsourced IC supply chain
  — Need for protection of chip design IP
- Logic locking, layout camouflaging, and split manufacturing

**Background**

Protect Your Chip Design Intellectual Property: An Overview

- Complex and globalized, outsourced IC supply chain
  - Need for protection of chip design IP
- Logic locking, layout camouflaging, and split manufacturing

Thank you!