Hardware Security and Logic Locking

Jonti Talukdar
ECE 538: VLSI System Testing
April 13, 2021
Modern Security Landscape

- **Network Security**
  1. Denial of Service (DoS),
  2. Man-in-the-Middle attacks,
  3. Spear phishing attacks,

- **Information Security**
  1. Corruption,
  2. Leakage,
  3. Unavailability,

- **Software Security**
  1. Software virus,
  2. File corruption,
  3. IP theft/piracy,
  4. Illegal access,

- **Hardware Security**
  1. Hardware Trojan attack,
  2. Hardware IP Theft,
  3. Reverse-eng/Cloning,
  4. Side-channel attacks,

- **Root of Trust**

---

[1]
Hardware Security

• Hardware Security focuses on attacks crafted to steal or compromise hardware assets and methods designed to protect these assets.

• Assets under consideration:
  1. Integrated Circuit IPs
  2. PCBs
  3. Information in the IPs:
     a. Sensitive User Data
     b. Cryptographic Keys
     c. Firmware
     d. Configuration Data

• Generally, attack impact and its difficulty changes with the attack surface.
Hardware Security vs Hardware Trust

- **Hardware Security** issues arise from that hardware's own vulnerability to different types of attacks at different levels (PCB, chip, IP, etc.)

- **Hardware Trust** arises from the involvement of untrusted entities in the hardware lifecycle itself. These entities include tool vendors, foundry, test and distribution facilities, CAD tools, etc.
Terminologies

• **Attack Vectors**: means or paths for bad actors (attackers) to get access to hardware components for malicious purposes, for example, to compromise it or extract secret assets stored in hardware.

• **Attack Surface**: is the sum of all possible security risk exposures. In the context of hardware security, attack surfaces define the level of abstraction in which the attacker focuses on launching a hardware attack.

  ➢ **Chip-level Attacks**: Cloning, counterfeiting, IP piracy, reverse engineering
  ➢ **PCB-level Attacks**: Optical inspection, X-Ray tomography, PCB tampering, bypassing
  ➢ **System-level Attacks**: Hardware-software interaction. E.g., DFT infrastructure at system level (JTAG), unauthorized control, access sensitive data.

Goal: Minimize Attack Surface, Thwart Attack Vectors
Terminologies

• **Threat Model**: Describes the threats as well as resources available to an adversary.

• **Trust Model**: Describes the trusted parties or components.

• **Vulnerability**: Weakness in the hardware architecture, implementation, or design/test process, which can be exploited by an attacker to mount an attack.

  ➢ **Functional Bug**: Arise from poor design/verification practice. These vulnerabilities might be discovered accidentally and hence vulnerable against malicious entities.

  ➢ **Side-Channel Bug**: Implementation level bug that leaks critical information stored in hardware component.

  ➢ **Test/Debug Infrastructure**: Complete controllability and observability of the design may provide unwanted control of system to an attacker.

  ➢ **Access Control / Information Flow Issue**: System may not distinguish between authorized and unauthorized users, giving attacker access to functionality that can be used to mount attacks.
Security in Chip Lifecycle

State of practice in security design and validation along the lifecycle of an SoC

Conflict between Security and DFT/Test/Debug
## Hardware Attacks: Categories

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hardware Trojans</strong></td>
<td>An attacker either in the design house or in the foundry may add malicious circuits or modify existing circuits.</td>
</tr>
<tr>
<td><strong>IP Piracy and IC overbuilding</strong></td>
<td>An IP user or a rogue foundry may illegally pirate the IP without the knowledge and consent of the designer.</td>
</tr>
<tr>
<td><strong>Reverse Engineering</strong></td>
<td>An attacker can reverse engineer the IC/IP design to his/her desired abstraction level. He can then reuse the recovered IP or improve it.</td>
</tr>
<tr>
<td><strong>Side-channel Analysis</strong></td>
<td>An attacker can extract the secret information by exploiting a physical modality (power, timing, EM) of the hardware that executes the target application</td>
</tr>
<tr>
<td><strong>IC Overbuilding</strong></td>
<td>A malicious foundry may build more than the required number of ICs and sell the excess ICs in the gray market.</td>
</tr>
<tr>
<td><strong>Counterfeiting</strong></td>
<td>An attacker illegally forges or imitates the original component/design.</td>
</tr>
</tbody>
</table>
Hardware Attacks: Threat Model

Attack goal
- Leak sensitive information
- Modify functionality
- Reduce reliability
- Deny Service
- Steal design
- Identify trade secrets

Attack
- Maliciously modify circuits (Trojan)
- Illegally copy & reproduce design (IP piracy)
- Reverse-engineering
- Analyze Side-channels
- Counterfeit

Attacker Location
- 3PIP vendor
- SoC integrator
- Foundry
- PCB assembly unit
- Test facility
- End user
- Recycling/repackaging facility
Hardware Attacks: Countermeasures

**IP Watermarking**
A designer’s signature is embedded into the design artifact.

**Hardware Obfuscation**
Obfuscation hides the functionality and implementation of a design by inserting additional gates into it.

**IC Metering**
Metering is a set of protocols and techniques, either passive or active, which assign a unique ID to each manufactured IC and help in tracking of individual ICs post-manufacturing.

**Split Manufacturing**
Design is split into two parts, corresponding to backend-of-the-line (BEOL) and front-end-of-the-line (FEOL) metal layers, that are manufactured in separate foundries and ultimately stacked together.

**Camouflaging**
Camouflaging is a countermeasure that aims at preventing reverse engineering carried out by end-users. It replaces selected gates in a design with their camouflaged counterparts.

[3]
Hardware Attacks: Countermeasures

- Attack
  - Hardware Trojans
  - IP piracy & IC overbuilding
  - Reverse engineering
  - Side channels
  - Counterfeit ICs

- Countermeasure
  - Design obfuscation
  - IP watermarking
  - IP fingerprinting
  - IC metering
  - Split manufacturing
  - IC camouflaging
  - IC information leakage reduction
  - Key-based authentication
  - Noise injection
  - Secure-scan
  - Physical unclonable functions/Unique ID(s)
  - Aging sensor

- Security metric
  - Number of attempts to find the secret (or exploit)
  - Number of collisions
  - Amount of information leakage
  - Probability of detection (related to false negative)
  - Probability of false positive
  - Inter-chip/inter-design (response) distance
  - Number of independent identifiers (IDs)/keys
  - Stability of intra-chip responses to challenges
Logic Locking

Logic locking / Logic Obfuscation refers to the insertion of specific logic in the design which only allows authorized users with the valid key to functionally access the design.

Logic Locking provides security against multiple untrusted environments.

Types of Logic Locking
1. **Combinational Logic Locking**
2. **Sequential Logic Locking**
3. **Scan Locking**

<table>
<thead>
<tr>
<th>DfTr technique</th>
<th>SoC integrator</th>
<th>Foundry</th>
<th>Test facility</th>
<th>End-user</th>
</tr>
</thead>
<tbody>
<tr>
<td>Watermarking [19, 20, 22, 33]</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Camouflaging [2, 3, 28, 30, 38, 47, 50]</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>Split manufacturing [17, 18]</td>
<td>X</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Metering (passive) [1, 25, 26]</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Logic locking [37, 39, 43, 51]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

✓ denotes that a technique can protect piracy conducted by an untrusted entity
✗ denotes the inability to defend against attacks

[3]
Logic Locking Manufacturing Flow

Phase 1: Locking during System Design

Phase 2: Interaction with untrusted Entities

Phase 3: Activation in a Trusted Setting
In combinational logic locking, key gates are inserted in the combinational cone of logic to achieve gate-level obfuscation of the design.

**Original Circuit:**
Combinational cone of logic requiring protection/obfuscation

**Key gates Inserted:**
Key gates inserted and the key value determined from the type of key gate inserted

**Circuit Resynthesis:**
IP functionality completely obfuscated due to bubble pushing / absorption
Attacks on Logic Locking

- **Attacker’s Objective:** Recover the key that locks the design. This will allow the attacker to

- **Categories of attacks on logic locking:**

  1. **Algorithmic Attacks:** These attacks exploit the algorithmic weaknesses of a logic locking algorithm to extract the secret key. For example, sensitization attack, SAT attack, Circuit Partitioning attack, etc.

  2. **Approximate Attacks:** Contrary to the exact attacks, the approximate attacks extract a netlist that is approximately the same as the original netlist, i.e., the netlist may produce an incorrect output for only a few input patterns.

  3. **Structural Attacks:** The fundamental principle of the structural/removal attacks is to bypass and/or remove the protection logic and isolate the functionally correct netlist.

  4. **Side Channel Attacks:** Side-channel attacks exploit the covert physical channels, such as power and timing, to leak the secret information
Threat Model for Logic Locking

• **Threat Model** for Oracle-based Attacks:
  ✓ Attacker has access to the reverse-engineered netlist (obtained from layout)
  ✓ Attacker has access to functionally operational oracle (procured from the market).

• Oracle-access allows the attacker obtain correct input/output response pairs for the IP under attack

• Reverse-engineered netlist allows attacker to analyze and bypass built-in defenses to unlock the IP

• Popular Oracle-based attack:
  ❖ SAT-attack (and its enhancements)
  ❖ Sequential SAT

• Popular Oracle-free attacks:
  ❖ Removal attacks
  ❖ Bypass attacks based on netlist-analysis, signal skew, etc.
  ❖ Redundancy-based attacks
Case Study: Sensitization based Attack

Attacker tries to sensitize the key inputs to the primary output of the circuit.
Strong Logic Locking: Interfering Key-Gates

An illustration of pairwise secure key gates in SLL. None of the key bits K1 or K2 can be sensitized to a primary output unless the value of the other key-bit is known, or its effect can be muted.

A circuit with 4 key gates and their interference graph
Strong Logic Locking: Interfering Key-Gates

**Goal:** Determine the secret key used for logic encryption

**Attacker has:**
- Locked netlist
- Functional IC (with embedded key)

**Attacker does:**
- Compute the attack patterns from the locked netlist
- Applies them on IC
- Infers key from responses
Combinational SAT Attack on Logic Locking

- **Attack Approach:** Eliminate keys in the search space, gradually converging to the correct key

- Identify Distinguishing Input Patterns or DIPs: Inputs for which two keys lead to \( \text{diff} = 1 \).

\[
\overrightarrow{X}_i^d \leftarrow A^\text{SAT} \left( C_{\text{scr}}(\overrightarrow{X}, \overrightarrow{K}_1, \overrightarrow{Y}_1) \land C_{\text{scr}}(\overrightarrow{X}, \overrightarrow{K}_2, \overrightarrow{Y}_2) \land (\overrightarrow{Y}_1 \neq \overrightarrow{Y}_2) \right)
\]

- Once DIP Identified, possible to eliminate at least one of the two keys by querying the oracle.

- In each iteration, all keys producing the same incorrect output will be eliminated.

<table>
<thead>
<tr>
<th>abc</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>Incorrect keys identified</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</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>1</td>
<td></td>
</tr>
<tr>
<td>001</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>1</td>
<td></td>
</tr>
<tr>
<td>010</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>1</td>
<td></td>
</tr>
<tr>
<td>011</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>1</td>
<td>iter 3: other incorrect keys</td>
</tr>
<tr>
<td>100</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>1</td>
<td></td>
</tr>
<tr>
<td>101</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></td>
</tr>
<tr>
<td>110</td>
<td>1</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>iter 1: k2</td>
</tr>
<tr>
<td>111</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>iter 2: k1</td>
</tr>
</tbody>
</table>

Miter Circuit

SAT Attack Flow

1. Construct miter
2. Locked netlist
3. SAT solver
4. Query functional IC
5. DIP exists?
   - No
   - Yes
5.1 Return correct key value
   - iter 1: k2
   - iter 2: k1
   - iter 3: other incorrect keys
Mitigating SAT Attack

- Total execution time of SAT attack is given by $T = \sum_{i=1}^{\lambda} t_i$

- Increase $\lambda$, the number of DIPs required for a successful SAT attack. This can be achieved by controlling the distinguishing ability of the DIPs.

- Increase $t_i$, the time required for individual attack iterations. This task can be accomplished by making use of circuit structures that are inherently hard to resolve for the SAT solvers.

- Point-function based methods rely on increasing $\lambda$. 
Point function-based Schemes: SAR Lock

Point function circuit checks for the input key and generates the flip signal only for the correct key, thereby producing the correct output.
Drawbacks of Point Function-based Schemes

1. **Output Corruptibility:** These schemes suffer from poor output corruptibility as only for a single key (out of all possible keys), is the output of the circuit corrupted.

2. **Removal Attacks:** Identifying the skew of nets in the design facilitates easy detection and bypassing of the flip signal.

\[
\begin{align*}
    s_{n-\text{AND}} &= \prod_{i=1}^{n}(0.5 + s_i) - 0.5 \\
    s_{n-\text{NAND}} &= 0.5 - \prod_{i=1}^{n}(0.5 + s_i) \\
    s_{n-\text{OR}} &= 0.5 - \prod_{i=1}^{n}(0.5 - s_i), \\
    s_{n-\text{NOR}} &= \prod_{i=1}^{n}(0.5 - s_i) - 0.5.
\end{align*}
\]
In sequential logic locking, the state transition graph of the original circuit is modified to add additional obfuscation states and state transitions. Only the correct traversal of unlocking state transitions will lead to correct initialization of the functional state.
Scan Obfuscation

Methods either prevent scan access or obfuscate scan shift data.

1. **Scan data obfuscation:** Flip the scan chain data by inserting either XOR gates, inverters, etc.

   ![Scan data obfuscation diagram]

2. **Scan data encryption:** Using on chip crypto hardware to encrypt scan chain data.

   ![Scan data encryption diagram]
3. **Scan Chain Ordering:** Dynamically reconfigure the order of scan out by splitting single scan chain

4. **Dynamically Obfuscated Scan:** Using on chip crypto hardware to encrypt scan chain data.
In scan locking, the access to the scan chain is restricted to prevent unauthorized parties from accessing the scan chains.
BISTLock: Using BIST for IP Protection

• Boolean Satisfiability attack (SAT attack)
  • Use SAT solvers to recursively refine key space

• Removal attack
  • Identify protection mechanism from their topological structure or boolean functionality

• Significant overhead
  • Up to 40% overhead in area, delay and power
  • SFLL [1] reports 2.16% area, 5.62% power and 5.38% delay to ARM Cortex-M0
    • Yet, require resynthesizing in integration
Built-in Self Test (BIST)
BISTLock

Original BIST

PRPG

Enable

PRPG

MUX

Select

CUT

MISR

Enable

Barrier FSM

BISTLock Controller

Reset

Key Sequence

Test Mode
Barrier FSM

• Reset into the first locked state
• Reaches functional mode at INIT when the correct 128-bit key sequence is given
• Goes into a loop of ‘blackhole’ states when an incorrect key is entered
• Blackhole loops confuse the attacker on whether the key entered is correct
Proposed Security Metrics

• Potency: ability to corrupt original functionality
  • Output error rate
  • Average Hamming distance

• Resilience: resilience to reveal secret key
  • Search time

• Stealth: similarity/difference with the rest of the circuit
  • Number of attempts needed to identify the protection circuit in the worst-case
Evaluation: Overhead

• Area – compared with an already BIST-ed design

• Power
  • No dynamic power overhead
  • Negligible static power overhead due to small area overhead

• Delay
  • No impact on critical path
  • A K-cycle booting delay for a K-bit key

<table>
<thead>
<tr>
<th>IP Name</th>
<th>Description</th>
<th># Gates</th>
<th>Area Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCM-AES</td>
<td>Crypto core</td>
<td>22,890</td>
<td>1.28%</td>
</tr>
<tr>
<td>SHA-3-LOW</td>
<td>Crypto core</td>
<td>25,109</td>
<td>1.00%</td>
</tr>
<tr>
<td>MIPS32R1</td>
<td>CPU</td>
<td>39,577</td>
<td>1.13%</td>
</tr>
<tr>
<td>SHA-3-HIGH</td>
<td>Crypto core</td>
<td>46,996</td>
<td>0.73%</td>
</tr>
<tr>
<td>AES</td>
<td>Crypto core</td>
<td>233,024</td>
<td>0.16%</td>
</tr>
<tr>
<td>Rocket Chip</td>
<td>SoC</td>
<td>871,407</td>
<td>0.11%</td>
</tr>
<tr>
<td>Average</td>
<td>—</td>
<td>—</td>
<td>0.74%</td>
</tr>
</tbody>
</table>
Limitations

- Low-cost solution to counter IP piracy based on BIST
- New metrics to quantify security strength
- Practically secure against known attacks

Benefits

- Sensitization-based removal attack against the input multiplexer
- Camouflage Barrier FSM and locking signals against such removal attacks
Summary

1. We obtained a high-level overview of the field of Hardware Security
2. We learnt about different types of threats/attacks arising from the IC product lifecycle and their corresponding countermeasures.
3. We performed a deep dive into logic locking and understood its benefits over other countermeasures.
4. We also looked at other types of hardware obfuscation including sequential locking and scan chain obfuscation.
References

5. J. Lee et al., “Securing Designs against Scan-Based Side-Channel Attacks”, TCAD 2007
6. M. Rahman et al., “Dynamically Obfuscated Scan for Protecting IPs Against Scan-Based Attacks Throughout Supply Chain”, VTS 2017
7. R. Karmakar et al., “On Securing Scan Obfuscation Strategies Against ScanSAT Attack”, ISQED 2020
8. S. Chen et al., “BISTLock: Preventing IP piracy using BIST”, ITC 2020
Questions?