Designing Trustworthy Hardware

Simha Sethumadhavan
Joint work with PhD student Adam Waksman
Computer Architecture and Security Technologies Lab
Columbia University
A Quiz

• What do these hardware components do?
  – Can you guarantee that these chips don’t have backdoors?

• Will you blindly trust hardware if you were buying this?
  – Military Equipment
  – Financial Sector

PROBLEM: Currently impossible to certify trustworthiness of HW.
Why is Hardware Vulnerable?

- Hardware is complex
  - OpenSPARC T2 code base larger than the Chrome code base
  - Chips unsurprisingly have unintentional bugs, e.g., Intel errata

- Hardware design resembles software design
  - Often include third-party IP components (ip-extreme.com)
  - Review of IP difficult because of intentional obfuscation

- Complexity, distribution increases risk of backdoors
  - More hands, easier to hide
  - Designs are crafted by globally distributed teams

- Creates a significant security vulnerability
  - Hardware is the root of trust; software builds on hardware
  - Attacks have been reported [The Hunt for the Kill Switch]
Concern in Military Circles

Defending a New Domain
The Pentagon’s Cyberstrategy
William J. Lynn III

In 2008, the U.S. Department of Defense suffered a significant compromise of its classified military computer networks in North Carolina. Computer networks themselves are not the only vulnerability. Software and hardware are at risk of being tampered with even before they are linked together in an operational system. Rogue code, including so-called logic bombs, which cause sudden malfunctions, can be inserted into software as it is being developed. As for hardware, remotely operated "kill switches" and hidden "backdoors" can be written into the computer chips used by the military, allowing outside actors to manipulate the systems from afar. The risk of compromise in the manufacturing process is very real and is perhaps the least understood cyberthreat. Tampering is almost impossible to detect and even harder to eradicate. Already, counterfeit hardware has been detected in systems that the Defense Department has procured. The Pentagon’s Trusted Foundries Program, which certifies parts produced by microelectronics manufacturers, is a good start, but it is not a comprehensive solution to the risks to the department’s technological base. Microsoft and other computer technology companies have developed sophisticated risk-mitigation strategies to detect malicious code and deter its insertion into their global supply chains; the U.S. government needs to undertake a similar effort for critical civilian and military applications.
Overview of Hardware Process

RTL Code → Design Synthesis → Physical Synthesis → Device Fabrication → Device Audit
Who and what are we trusting?

Designers/IP Providers
- RTL Code
- Design Synthesis
- Physical Synthesis

Compiler Tools
- Device Fabrication

Foundry
- Device Audit

Untrusted Designer (Human/Insider)
- Waksman & Sethumadhavan, 2010
- Hicks et al., 2010
- Waksman & Sethumadhavan, 2011
- Waksman et al., 2013

Untrusted Compiler (Software)
- Kastner et al., 2010
- Smith et al., 2010
- Potkonjak et al., 2010
- Potkonjak et al., 2009
- Koushanfar et al., 2007

Untrusted Fabrication (Physical Process)
- Banga et al., 2008
- Chakraborty et al., 2008
- Agrawal et al., 2007
- Lee et al., 2004
- Quisquater et al., 2001
Taxonomy of Hardware Backdoors

• Backdoor = Trigger + Payload
  • Trigger: Mechanism for initiating an attack
  • Payload: Malicious, illegal action

• Why do we need a Trigger?
  
  • Trigger-less designs will be caught during validation
  
  • Most designs deploy intensive random transactional testing
    • Small units are validated thoroughly, followed by aggregations
    • Typically smaller units are validated for $10^6 - 10^8$ cycles
    • Larger units validated for fewer cycles
How many ways can a backdoor be triggered?

Triggers are finite state machines
  - Can change state only when time or input data changes

A complete taxonomy of hardware backdoors
Taxonomy of Attacks: Triggers

• Data Triggers
  – Cheat codes (CC)
  – Triggered by special instructions or data

• Pros/Cons of CC’s
  – Easy to bypass validation
    • 1 in $2^{64}$ chance!
  – However, hacker needs access to the machine
Taxonomy of Attacks: Triggers

- **Time Triggers**
  - *Ticking Timebomb (TT)*
  - Triggered over time

- **Pros/Cons of TTs**
  - **Easy to bypass validation**
    - 48-bit counter takes ~20 minutes @ 1 GHz
  - Easy to hide
  - However, open to everyone
Taxonomy of Attacks: Payloads

Is the payload separate from the normal instructions?

- **Emitter Attacks**
  - Extra malicious events
  - Separate from normal events
  - Unlikely to be noticed by user

- **Corrupter Attacks**
  - No extra malicious events
  - Normal operations altered
  - Difficult to engineer
Taxonomy of Attacks: Summary

For more elaborate taxonomy including Analog Backdoors Refer to Karri et al.
Common Solutions are Unsatisfactory

- Careful audits
  - Audits not completely effective at catching unintentional bugs
  - Can audits catch intentional, hidden backdoors?

- Random validation
  - Catching a 48-bit TT requires 281.4 trillion cycles of validation!
  - The chance of catching a 48-bit CC is $3.5 \times 10^{-15}$

- Static verification
  - Attacker has complete access to all design specifications
  - Attacker can work around theorems or proofs

- Cannot fix problem in run time software
  - All software runs on hardware
  - Software fix will likely use malicious hardware
Design (HDL) Security

- Static Identification (Hicks et al. 2010)
  - Software TCB
  - False positives (can be intentional)

- Method
  - Identify ‘quiescent’ circuits
  - Replace them with exception handlers
  - Exceptions are handled in software

- Infinite set of false negatives (Sturton et al. 2011)
IC Fingerprinting (Agarwal et al., 2007)

- Detects aberrant power-leakage information
- Requires ‘golden’ model, destructive testing
- Easy to catch complex trojans
- AES example: static power leakage from dormant backdoor
More Foundry-level Defenses

• Region-Based Identification (Banga et al., 2008)
  – Maximize in-region switching
  – Minimize out-region switching

• Physical trojan activation (Tehranipoor et al., 2009)
  – Several assumptions
  – Trojans off critical path
  – Latency resilience for additional flip-flops
  – Trojans have reasonably high activation likelihood
  – Trojans are data-sensitive but time-insensitive

• Pre-requisites
  – Reliable scan chains
  – “Golden” models
One Solution in Detail

• Silencing Hardware Backdoors
  – Waksman and Sethumadhavan
Solution: Obfuscation of Inputs

Backdoor = **Trigger** + **Payload**

**Inputs**

- **Hides Triggers**

**Outputs**

- **Deliver Payload**

![Logic diagram of 8-input digital multiplier](logic_diagram.png)
Solution: Obfuscation of Inputs

Backdoor = \textbf{Trigger} + \textbf{Payload}
Taxonomy of Interfaces

Global Control Data Test

HARDWARE MODULE

Outputs
Three Kinds of Backdoor Triggers

Global

Control

Data

HARDWARE MODULE

Outputs
Three Kinds of Backdoor Triggers

- Global
- Control
- Data

HARDWARE MODULE

Outputs
Three Kinds of Backdoor Triggers

Global

Control

Data

HARDWARE MODULE

Outputs
Three Interfaces, Three Triggers

- **Global**
- **Control**
- **Data**

**Global**: Timebombs

**Control**: Cheatcodes
- Single-shot
- Sequence

**Data**: Cheatcodes
- Single-shot
- Sequence
Three Kinds of Backdoor Triggers

Global

Control

Data

HARDWARE MODULE

Outputs
Three Interfaces, Three Triggers

- **Global**
  - Ticking

- **Control**
  - Cheatcodes
    - Single-shot
    - Sequence

- **Data**
  - Cheatcodes
    - Single-shot
    - Sequence

> Timebombs
Solution: Obfuscation of Inputs

Backdoor = Trigger + Payload
Three Solutions For Three Triggers

• **Goal:** Obfuscate information coming into each interface

  • **Ticking timebombs**
    • Periodically reset the power

  • **Single-shot Cheatcodes**
    • Encrypt data values

  • **Sequence Cheatcodes**
    • Reorder events or insert dummy events
Power Resets

• Power to modules is reset periodically
  – Time period = N – K cycles
  – N = Validation epoch
  – K = Time to restart module operation

• Forward progress guarantee
  – Architectural state must be saved and restored
  – Microarchitectural state can be discarded (low cost)
    • e.g., branch predictors, pipeline state etc.,
Power Resets: Security Analysis

Can trigger be stored to architectural state and restored later?

• No: Unit validation tests prevent this

• Reasons for trusting validation epoch
  – Large validation teams
  – Organized hierarchically

Can trigger be stored in non-volatile state internal to an unit?

• Maybe, but non-volatile state can be detected or avoided
Data Obfuscation
Data Obfuscation

- Homomorphic computation (Gentry 2009)
  - Data is operated on while encrypted
Data Obfuscation: Simple Case

Non-Computational Case

Router, Interconnect, Memory, Cache, Comparator, Buffer, Register, Table, etc.
Data Obfuscation: Simple Case

Non-Computational Case

Router, Interconnect, Memory, Cache, Comparator, Buffer, Register, Table, etc.
Data Obfuscation: Simple Case

Non-Computational Case

Router, Interconnect, Memory, Cache, Comparator, Buffer, Register, Table, etc.

Memory Controller

Store 5 to address 7
Data Obfuscation: Simple Case

Non-Computational Case
Router, Interconnect, Memory, Cache, Comparator, Buffer, Register, Table, etc.

Memory Controller
Data Obfuscation: Complex Case

- Homomorphic computation (Gentry 2009)
  - Data is operated on while in encrypted form
Data Obfuscation: Complex Case

- Homomorphic computation (Gentry 2009)
  - Data is operated on while in encrypted form

- General solution is expensive
  - Requires special hardware solutions
  - Examples: Adders, multipliers, RSA crypto units
Sequence Breaking

• Prevent sequences from being predictable by the user
  • Pseudorandom reordering of events

New Module
Sequence Breaking

- Prevent sequences from being predictable by the user
  - Pseudorandom reordering of events
Sequence Breaking

- Prevent sequences from being predictable by the user
  - Insert events when correctness conditions prevent reordering

New Module
Sequence Breaking

- Prevent sequences from being predictable by the user
  - Insert events when correctness conditions prevent reordering

- Works for finite sets (not just ordered sequences)

Set is large enough for attacker to use $\leftrightarrow$ Set is large enough for validation engineer to catch
However, duplication is prohibitively expensive

- Non-recurring design, verification costs due to duplication
- Recurring power and energy costs
Open Problems

• Randomized triggers
  • Determine the level of threat from randomized backdoors
    • RNGs, other true sources of randomness
    • Uncontrolled payloads at uncontrolled times

• Secure usage of non-volatile memory technologies
  • Incorporate non-volatile memory in a trusted way
    • Improvements to and increased use of Flash
    • PCM and other new technologies
Open Problems

• More efficient homomorphic functions
  • Efficient obfuscation for computational units
    • Units classified by type
    • Formal understanding of costs

• Automated implementation of backdoor protection
  • Compiler and/or language additions
    • Tools for designers
    • Simple language constructs for HDLs
Summary

- Prevent the triggering of hidden backdoors
  - Hardware-only solution
    - Low performance impact
    - Low power/area overhead
  - Prototype results demonstrate low overhead
    - IEEE Design and Test, May 2013
  - Revealed new open problems
    - Challenges for processors/embedded systems
    - Linguistic challenges
  - Vastly raises the bar against hardware backdoors


What we have not covered

• Backdoor protection
  – Methods for catching fabrication backdoors
  – Online bibliography

• Physical Unclonable Functions

• Hardware metering
Hardware’s Role in Security

• Security is a full-system property
  – Both hardware and software need to be secure
  – Traditionally, hardware hasn’t received as much attention

• Hardware focus changes notions of what’s “possible”

• Attacks
  – Hardware is the root of trust
  – Compromised hardware difficult to detect thru software

• Defenses
  – Hardware mechanisms offer smaller attack surface
  – Energy-efficient
SPARCHS Project at Columbia

- Security as a first-order design requirement

- Doctrine: Security must be designed from hardware up for practical, usable secure systems

- Contributions

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Instruction Set Randomization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Information Flow Tracking in SoCs</td>
</tr>
<tr>
<td></td>
<td>Security-aware HW/SW interface design</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Microarchitecture</th>
<th>Support for Migration, repl. execution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Malware detection (ISCA 13)</td>
</tr>
<tr>
<td></td>
<td>Side-channel Mitigation: Timewarp (ISCA12)</td>
</tr>
<tr>
<td></td>
<td><em>Side-channel Measurement: SVF (ISCA 12)</em></td>
</tr>
</tbody>
</table>

| Hardware Design                 | Hardware Backdoors (Oakland 10,11, DT 13) |