Introduction to Hardware Security

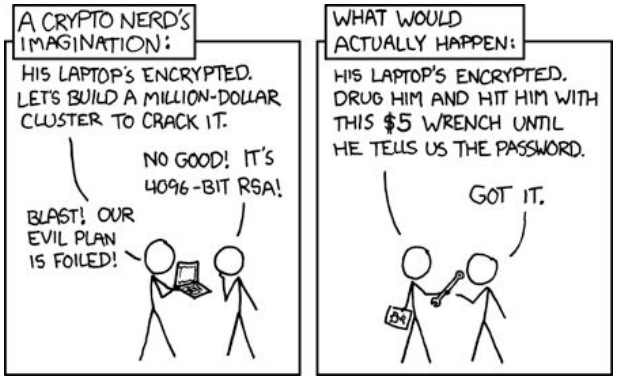
Prof. Marco Ottavi

Ing. Alessandro Palumbo

Hardware Security Problem

Cybersecurity experts have traditionally assumed that the hardware underlying information systems is secure and trusted. However such assumption is no longer true.

Hardware cannot be considered the root of trust



Hardware Security and Hardware Trust

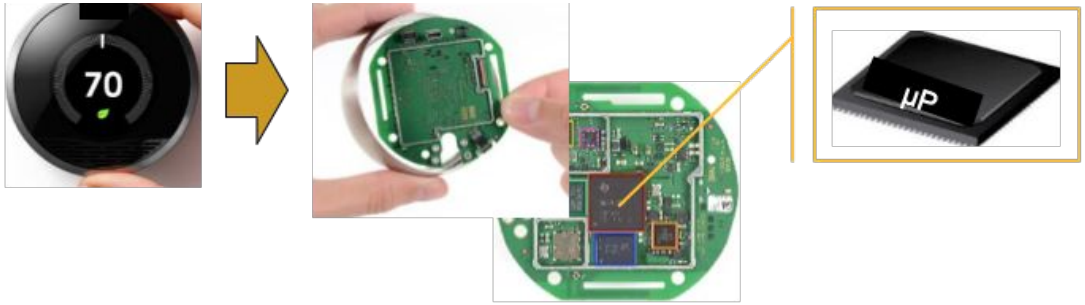
Two main problems with the root of trust assumption

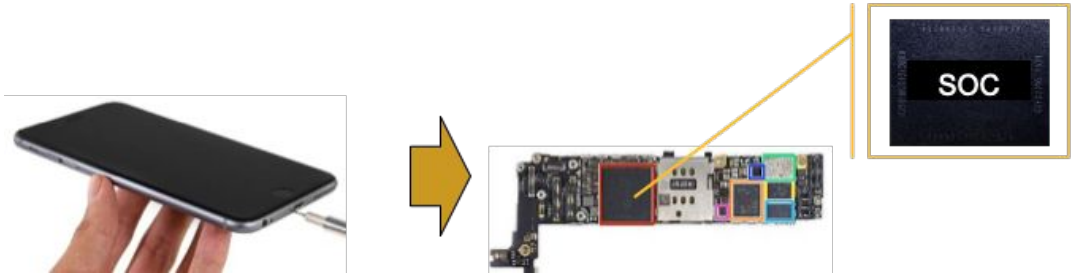
1) **Hardware Trust**: are we sure that the hardware has been made by trusted sources?
   a) are there any unwanted malicious modifications?
   b) These modifications are in general considered Trojans and can have different effects
2) **Hardware Security**
   a) are there any vulnerabilities that could be exploited by an attacker?
   b) Architectural vulnerabilities (example Spectre, Meltdown)
   c) Side Channel Attack vulnerabilities

Definitions

- **Threat**: Set of circumstances that has the potential to cause loss or harm
- **Vulnerability**: Weakness in the secure system
- **Attack**: The act of a human exploiting the vulnerability in the system
- **Computer security aspects**
  - **Confidentiality**: the related assets are only accessed by authorized parties
  - **Integrity**: the asset is only modified by authorized parties
  - **Availability**: the asset is accessible to authorized parties at appropriate times

What is Hardware?

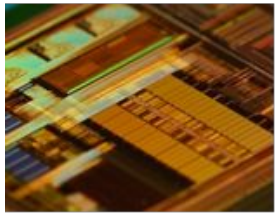


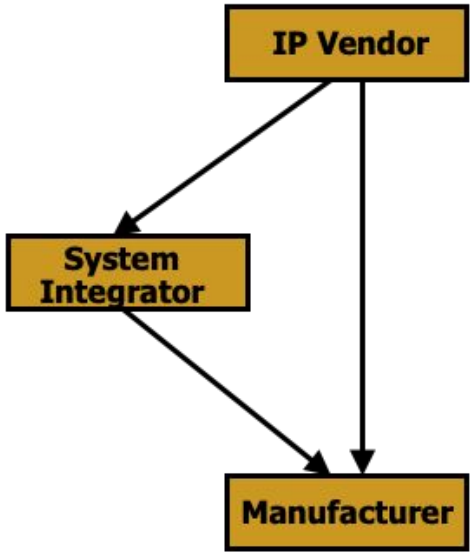


Electronic System

System Hardware – acts as the **“root-of-trust”**: PCB → IC (SoC | μP)

Hardware Threats





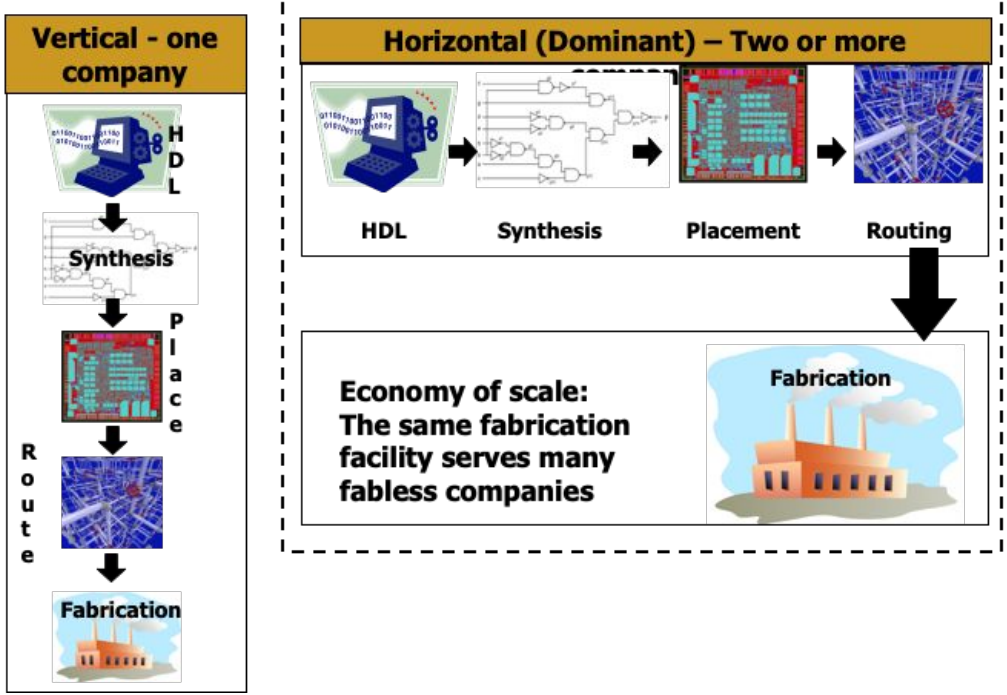
Any of these steps can be untrusted

IP: Intellectual properties sometimes provided by third party vendors

System Integrator combines several IPs into a chip design

Manufacturer fabricates the chips based on the received design

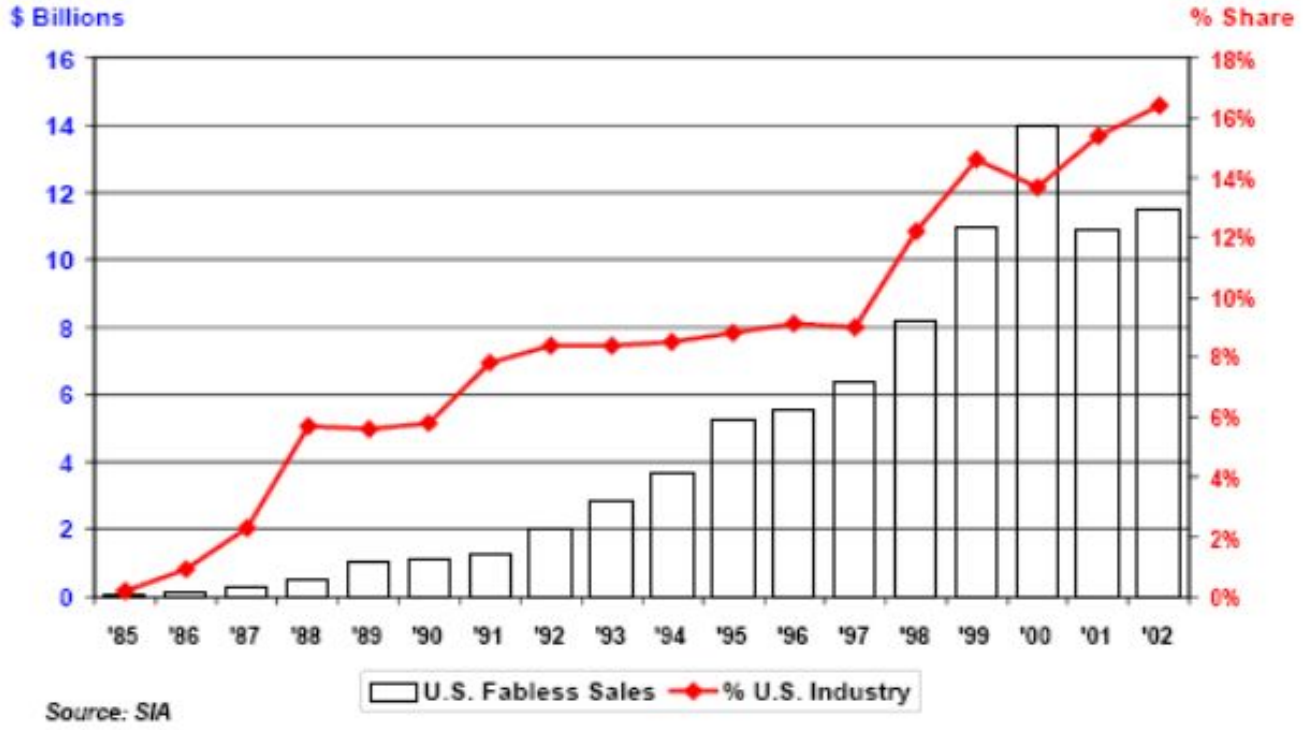
VLSI Industry: Business Model



Vertical Model: all in-house development → high costs, low economy of scale

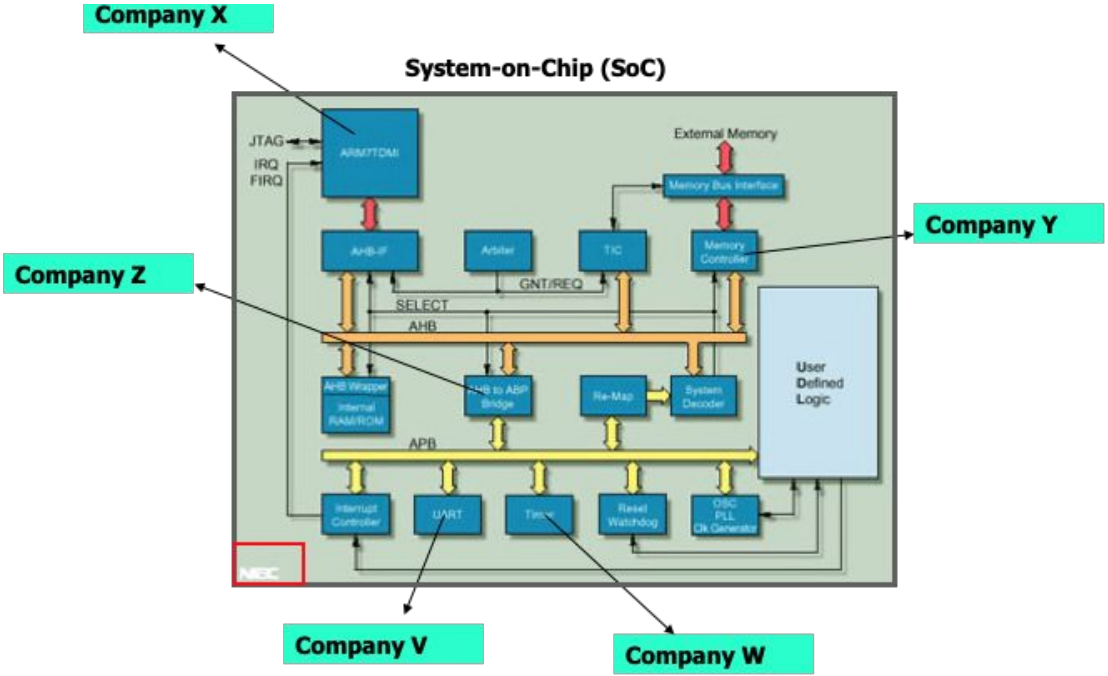
Horizontal Model: several companies involved -->lower costs, economy of scale

Fabless industry up to 16% of U.S. Chip industry



Source: SIA

Issues with Using Third Party IP



A modern System on Chip could be composed of IP provided by several vendors

These companies are located across the world There is no control on the design process

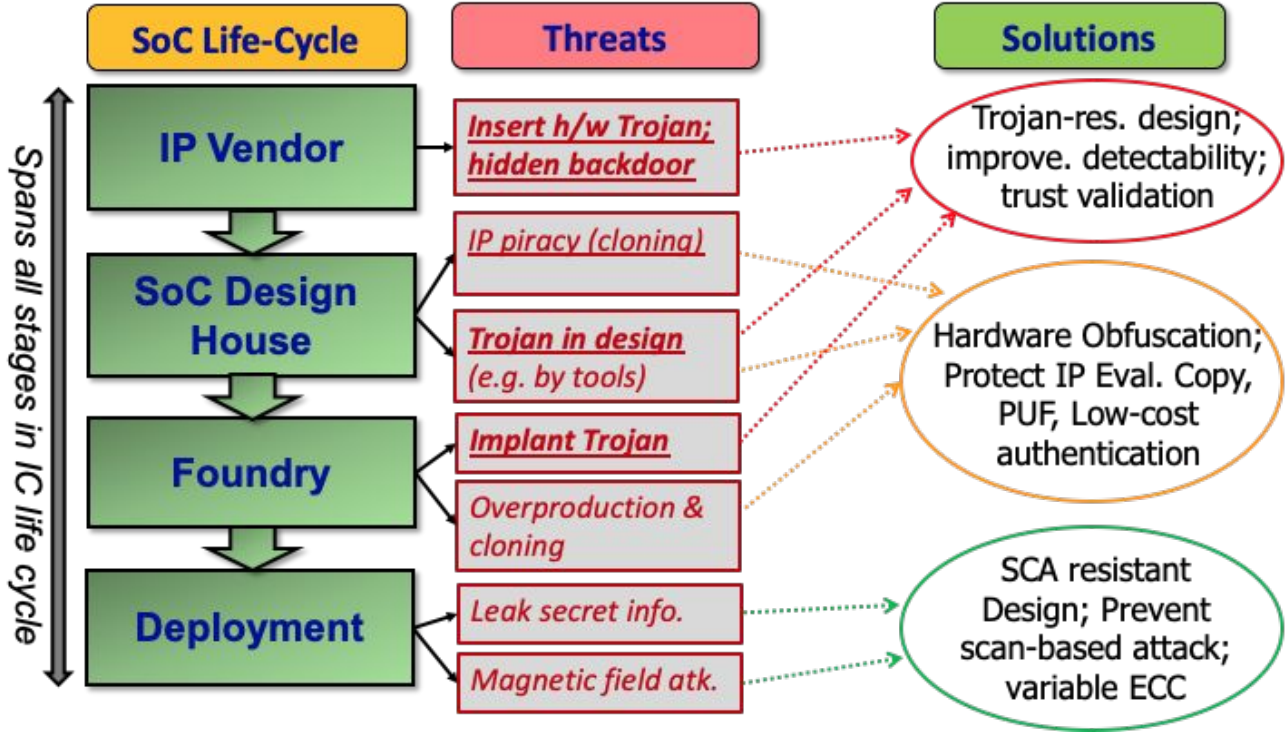
Where are the modern chips developed?

Throughout the globe

The phases of design, manufacturing, testing, packaging are a truly global activity

This raises potential trust issues

Hardware Threats



Threats can show up at different levels of the development of a Soc.

Countermeasures must be taken into account throughout the process

Hardware Vulnerabilities

Hardware can be affected by several Vulnerabilities:

- **Physical Attacks** (e.g. side channel attacks; microarchitectural vuln.)
- **Trojan Horses** (implemented at different design levels)
- **IP Piracy** (cloning of IP)
- **IC Piracy & Counterfeiting** (cloning, overproduction)
- **Backdoors** (modifications leaking secret)
- **Tampering** (e.g. FPGA bitstream modifications)
- **Reverse Engineering**

Adversaries

- Individual, group o foreign governments
  - Pirating the IPs – illegal use of IPs
  - Inserting backdoors, or malicious circuitries
  - **Implementing Trojan horses**
  - Reverse engineering of ICs
  - **Spying by exploiting IC vulnerabilities**
- System integrators
  - Pirating the IPs
- Fabrication facilities
  - Pirating the IPs
  - Pirating the ICs
- Counterfeiting parties
  - Recycling, cloned, etc.

So, what do we focus on?

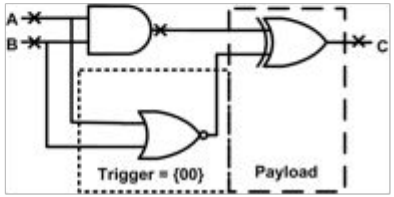
- **Hardware Trust Issues**
  - Hardware Trojans
  - HWT Classification and Detection Approaches
- **Hardware Security Issues**
  - Side Channel Attacks
  - Attacks exploiting vulnerabilities of the Hardware Implementation

What is Hardware Trojan?

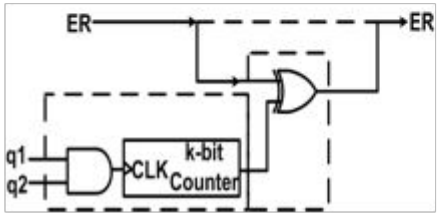
- **Hardware Trojan:**
  - A malicious addition or modification to the existing circuit elements.
- **What hardware Trojans can do?**
  - Change the functionality
  - Reduce the reliability
  - Leak valuable information
- **Applications that are likely to be targets for attackers**
  - Military applications
  - Aerospace applications
  - Civilian security-critical applications
  - Financial applications
  - Transportation security
  - IoT devices
  - Commercial devices
  - More

HW Trojan Examples and Models

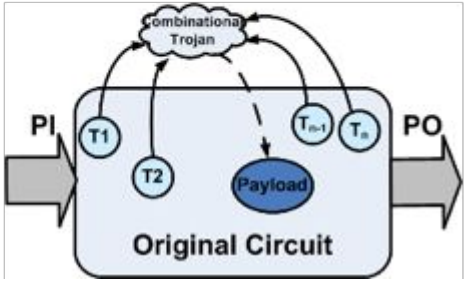
Comb. Trojan Example



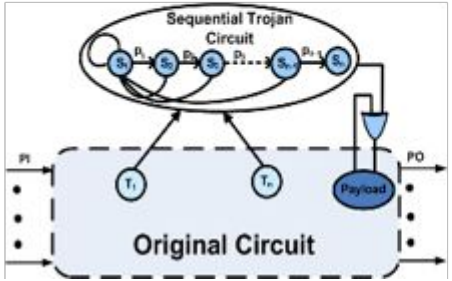
Seq. Trojan Example



Comb. Trojan model



Seq. Trojan Model

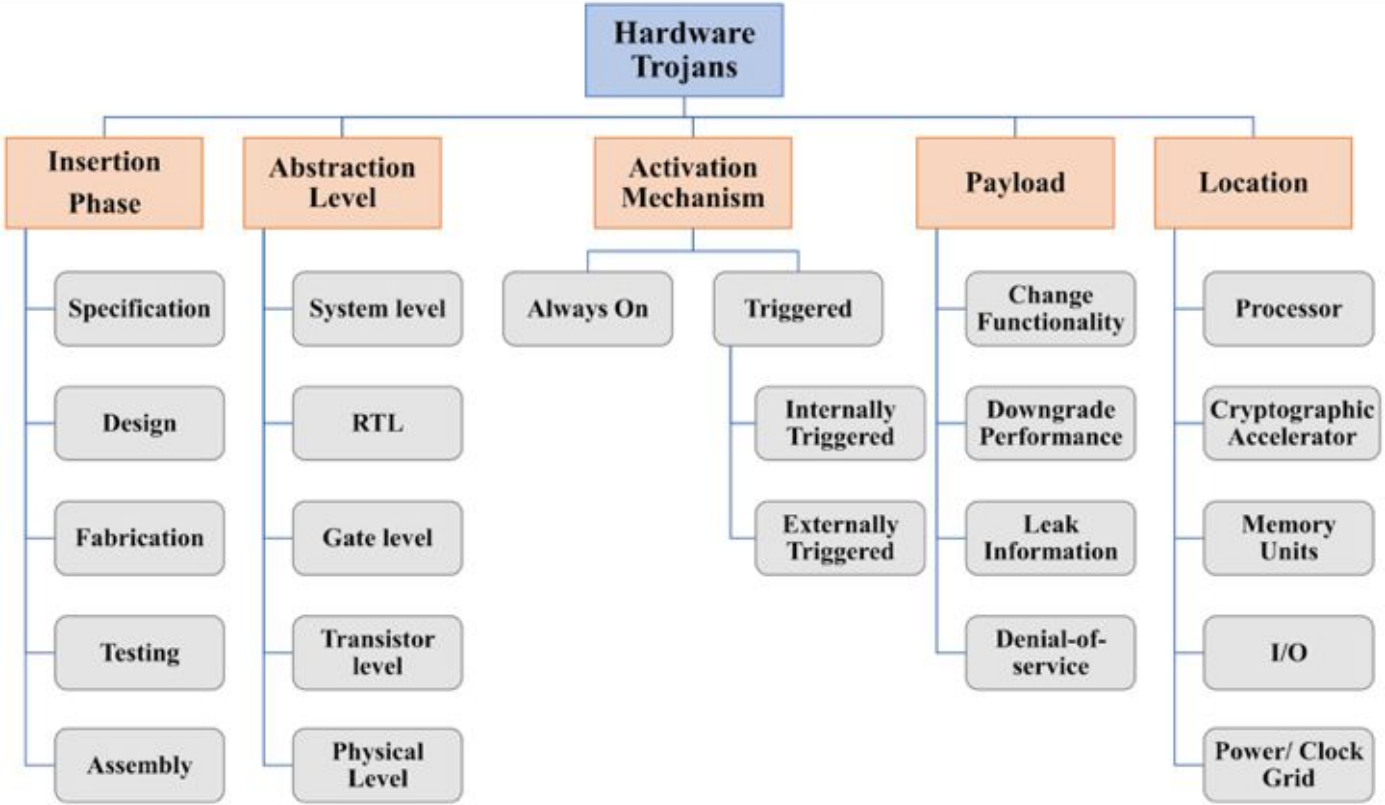


Two simple examples of HWT inserted in a digital circuit

Combinatorial: the Trojan flips the output only if the inputs are in 00

Sequential: the Trojan is a counter clocked by the values of q1 and q2 → flips output at a certain value

Trojan Taxonomy



Trojans can be classified based on several metrics

Insertion phase

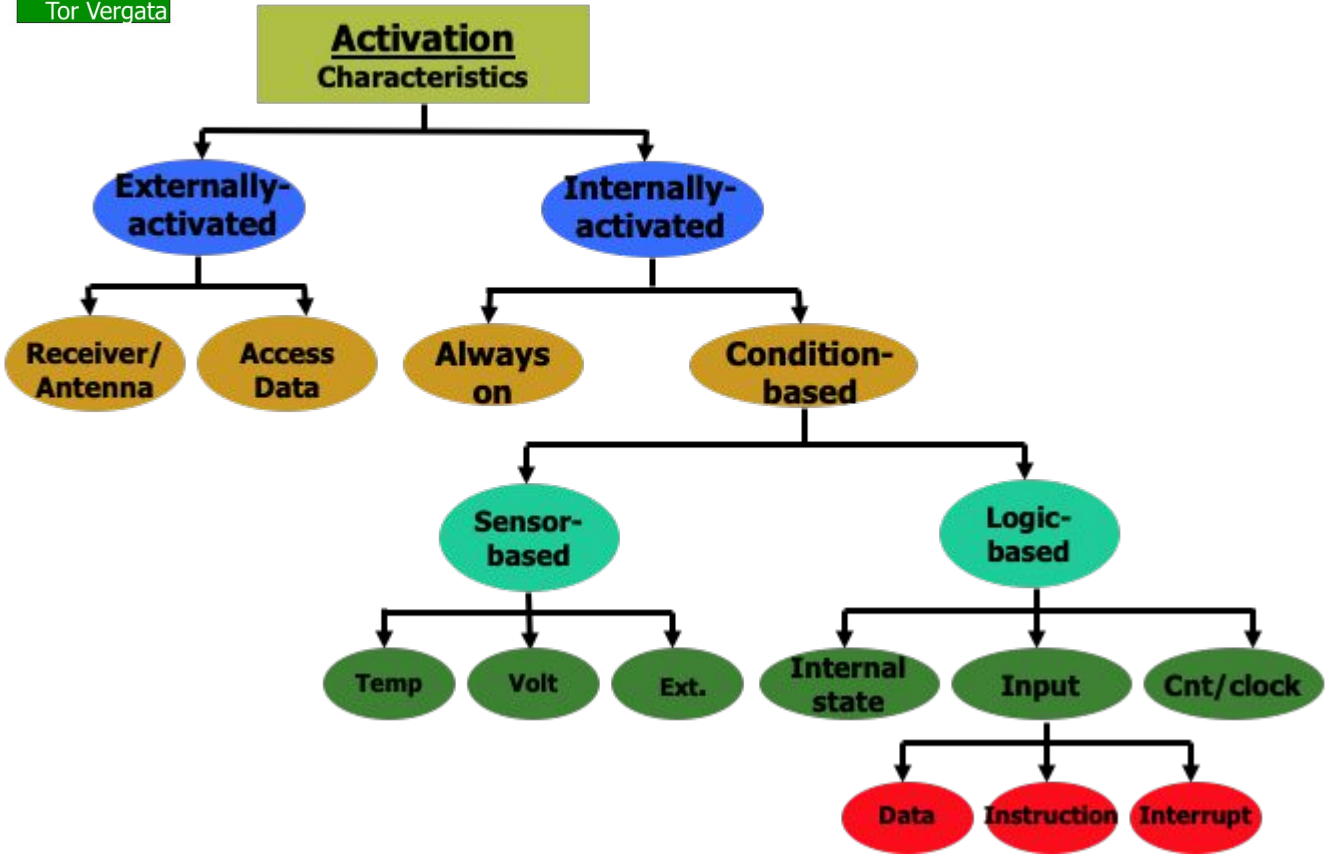
Abstraction level

Activation Mechanism

Payload

Location

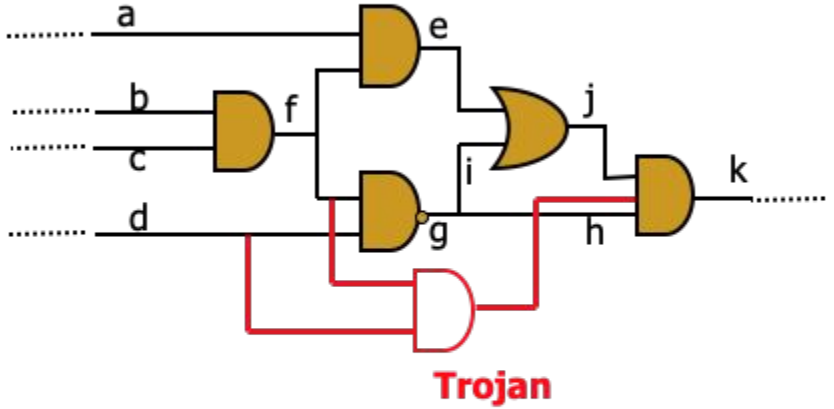
Trojan Activation Classification



Activation of Trojans can be based on several metrics

Countermeasures must take into account the presence of different activation approaches!

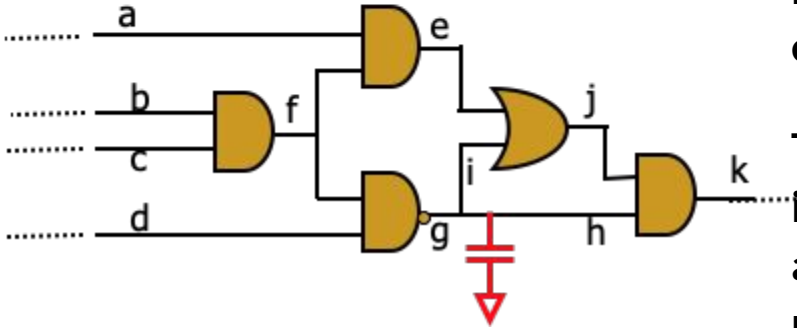
Example of two payloads



Modify Function

Modify Specification:

Noise, Delay and Temperature



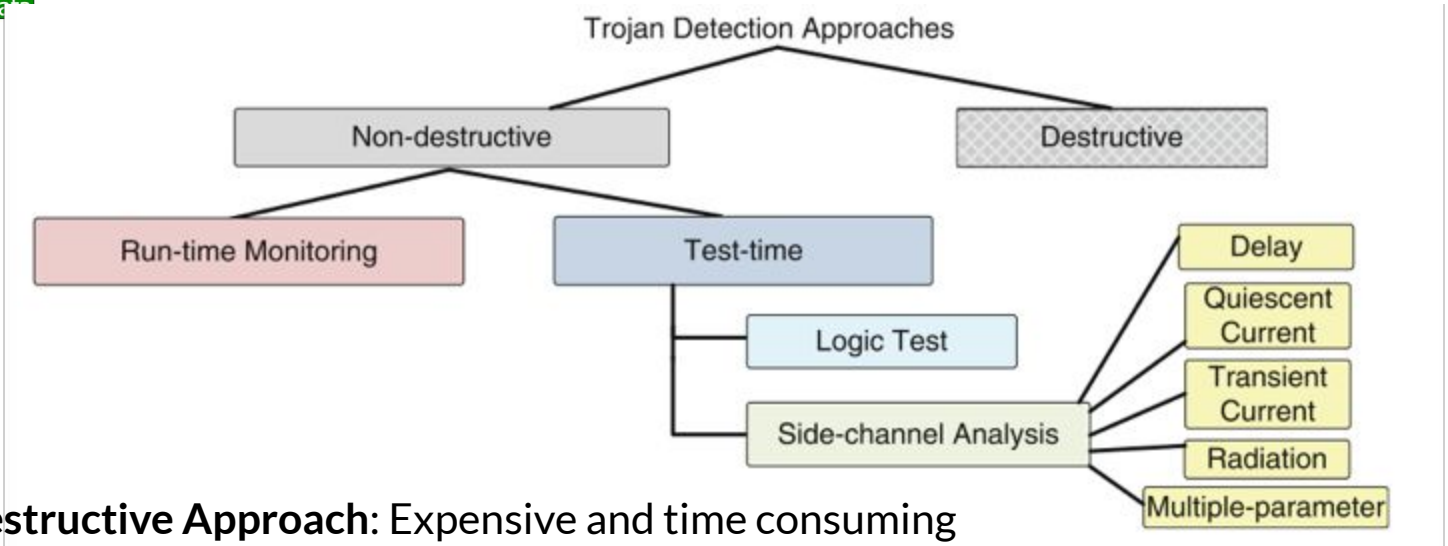
Trojan payload defines its ability to change the normal behavior

Two examples

Trojan changes the output by gating the output when either d or f are =0

Trojan changes delay thus increasing critical path by adding capacitive load on node g

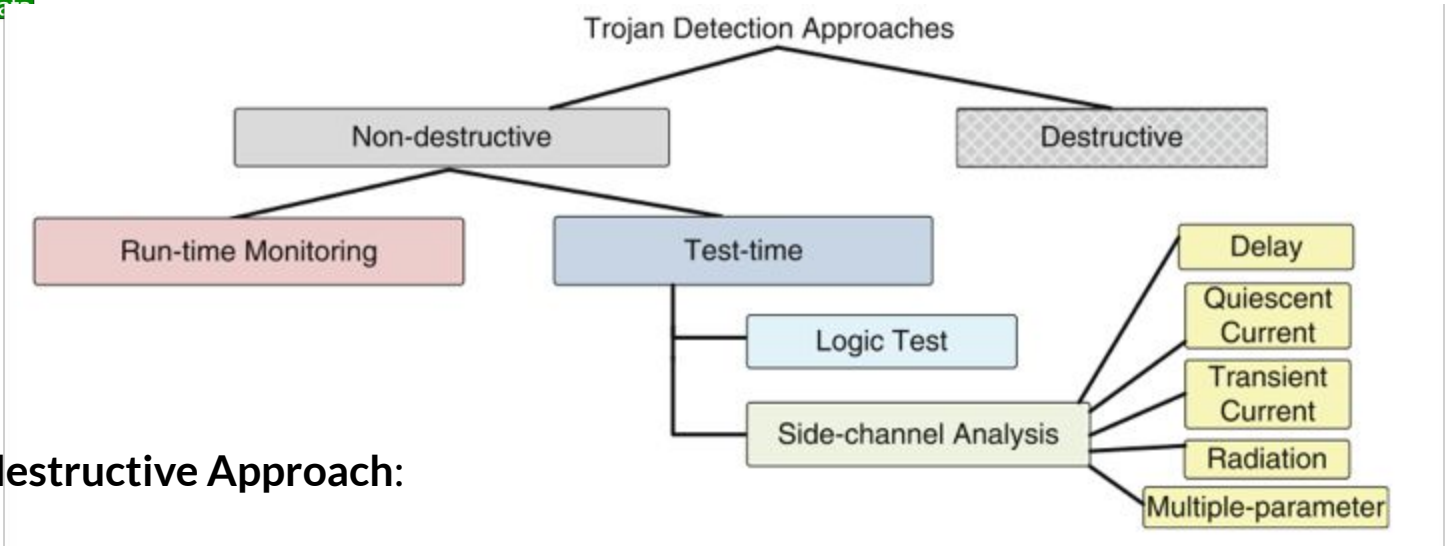
Classification of Trojan Detection Approaches



Destructive Approach: Expensive and time consuming

- Reverse engineering to extract layer-by-layer images by using delayering and Scanning Electron Microscope
- Identify transistors, gates and routing elements by using a template-matching approach – **needs golden IC/layout**

Classification of Trojan Detection Approaches



Non-destructive Approach:

- **Run-time monitoring**: Monitor abnormal behavior during run-time
  - Exploit pre-existing redundancy in the circuit
  - Compare results and select a trusted part to avoid an infected part of the circuit.
- **Test-time Authentication**: Detect Trojans throughout test duration.
  - Logic-testing-based approaches
  - Side-channel analysis-based approaches

Hardware Trojan Benchmarks

- A set of **trust benchmarks** for researchers in academia, industry, and government is needed to
  - Provide a baseline for examining diverse methods developed
  - Establishing a sound basis for the hardness of each benchmark instance
  - Help increase reproducibility of results by others who intend to employ certain methodologies in their design flow
- See NSF supported **Trust-Hub** website (www.trust-hub.org)
  - Complete taxonomy of Trojans
  - More than 120 trust benchmarks available which were designed at different abstraction levels, triggered in several ways, and have different effect mechanisms
  - More than 300 publications used these benchmarks

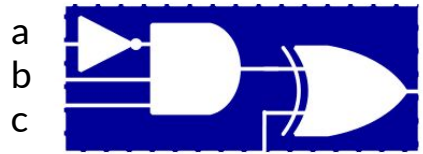
Runtime monitoring

Can we add specific redundancies to the project to detect malicious behaviour?

- At design phase include monitors (later on this)
- Can be used for random and malicious error detection
  - Soft/firm IP trojan detection in FPGA
  - Hard IP Trojan detection on SoC
- On microprocessor architectures leverage new available open ISA RISC-V

Logic Testing Approach

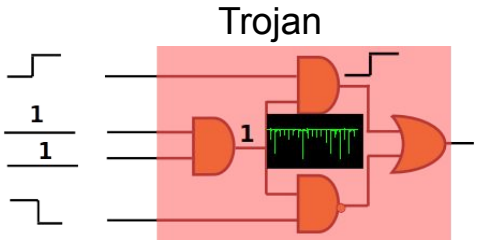
- **Logic-testing approach** focuses on test-vector generation for
  - Activating a Trojan circuit
  - Observing its malicious effect on the payload at the primary outputs
  - Both functional and structural test vectors are applicable.
- **Pros & Cons:**
  - **Pros**:
    - Straight-forward and easy to differentiate
  - **Cons**:
    - The difficulty in exciting or observing low controllability or low observability nodes.
    - Intentionally inserted Trojans are triggered under rare conditions. (e.g., sequential Trojans)
    - It cannot trigger Trojans that are activated externally and can only observe functional Trojans.



Example: Find vector to activate Trojan Trigger Condition
(a=0, b=1, c=1)

Side-Channel Analysis Approach

- All the side-channel analyses are based on observing the effect of an inserted Trojan on a physical parameter such as
  - **IDDQ**: Extra gates will consume leakage power.
  - **IDDT:** Extra switching activities will consume more dynamic power.
  - **Path Delay**: Additional gates and capacitance will increase path delay.
  - **EM:** Electromagnetic radiation due to switching activity
  -
- **Pros & Cons**
  - **Pros:** It is effective for Trojan which does not cause observable malfunction in the circuits.
  - **Cons**: Large process variations in modern nanometer technologies and measurement noise can mask the effect of the Trojan circuits, especially for small Trojan.



Example: detect power, delay, EM emission fingerprints of the inserted Trojan

NEEDS GOLDEN CHIP

Summary of Trojan Detection Techniques

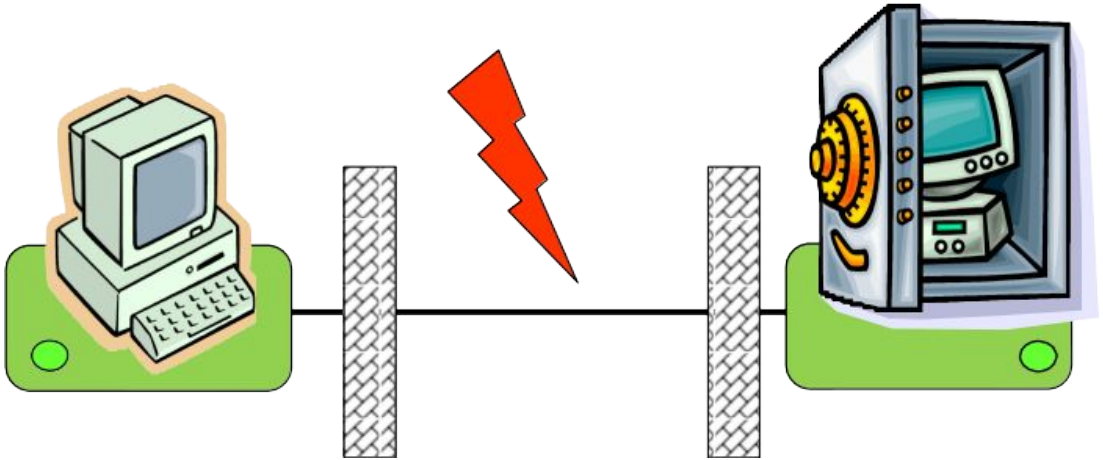
Trojan				Power Analysis	Delay Analysis	Fully Activation
Trojan Classification	Physical Characteristics	Type	Functional	D	P	P
			Parametric	P	D	P
		Size	Small		D	P
			Large	D	P	P
		Distribution	Tight	D	D	P
			Loose	P	D	P
		Structure	Modify Layout	P	D	
	Activation Characteristics	Always-on			D	
		Condition-based	Logic-based	D	P	P
			Sensor-based	D		
	Action Characteristics	Modify Function		D	P	
		Modify Spec.	Defects	P	D	P
			Reliability	P	P	P

P: Detection is possible D: High level of confidence

What about Side Channel Attack?

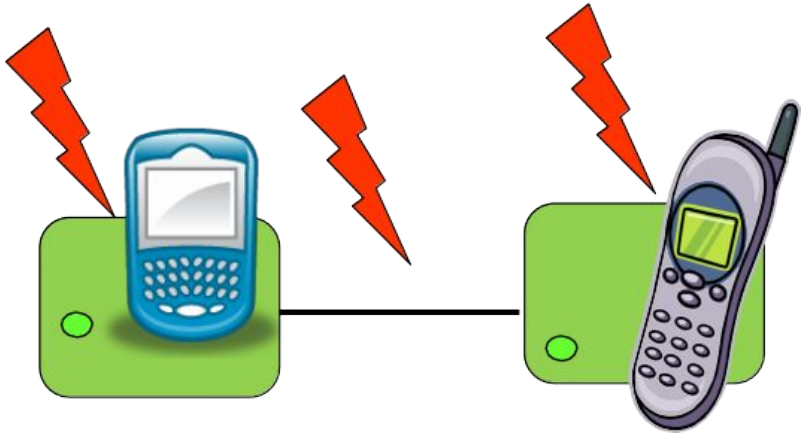
- Classic cryptography views the secure problems with **mathematical abstractions**
- The classic cryptanalysis has had a great success and promise
  - Analyzing and quantifying crypto algorithms' resilience against attacks
- Recently, many of the security protocols have been attacked through **physical attacks (also known as Side Channel Attacks)**
  - Exploit weaknesses in the cryptographic system hardware implementation aimed to recover the secret parameters

Traditional Model (simplified view)



- Attack on channel **between communicating parties**
- Encryption and cryptographic operations in **black boxes**
- Protection by strong mathematical algorithms and protocols
- Computationally secure

What is happening now (IoT and Embedded)



- New Model (also simplified view):
  - Attacks on **channel and endpoints**
  - Encryption and cryptographic operations in **gray boxes**
  - Protection by strong mathematical algorithms and protocols
  - **Protection by secure implementation**

- **Need secure implementations not only algorithms**

Where are the weaknesses?

A system is as secure as its weakest link

What is Side-Channel Attack?

- Side-Channel attacks aim at **side-channel inputs and outputs,** bypassing the theoretical strength of cryptographic algorithms → use additional information to break the cryptographic protection

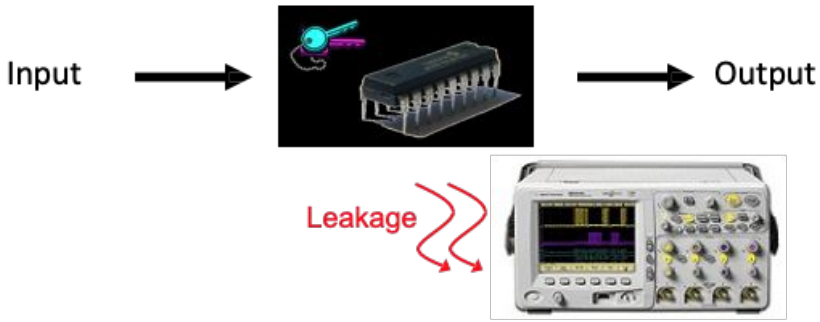
- Five commonly exploited side-channel emissions:
  - Power Consumption
  - Electro-Magnetic
  - Optical
  - **Timing and Delay**
  - Acoustic

Side Channel information Leakage

Physical attacks ≠ Cryptanalysis

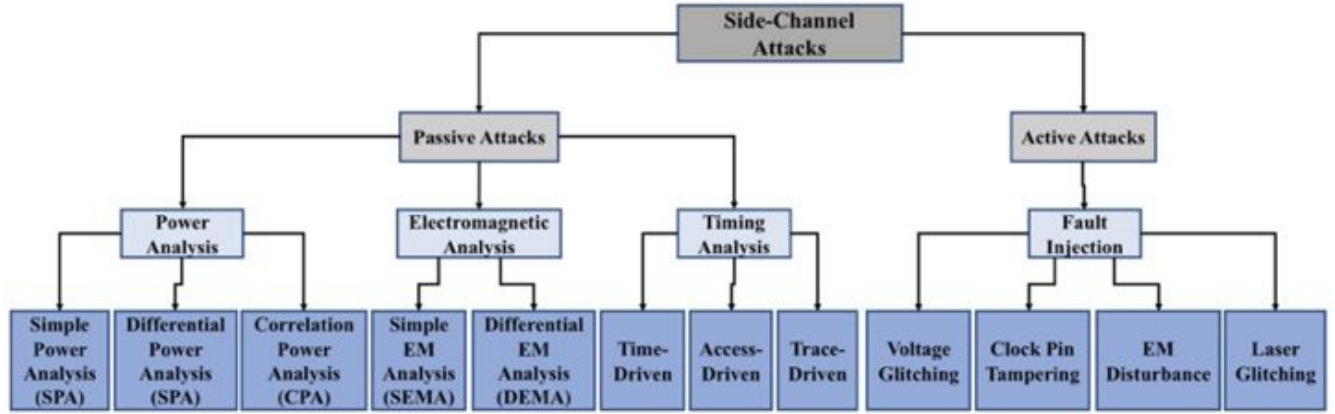
(gray box, physics) (black box, maths)

- Does not tackle the algorithm's math



Observe **physical quantities in the device's vicinity** and use additional information during cryptanalysis

Taxonomy of Side Channel Attacks



Passive attacks: observe the behavior of the device to infer information about the secret

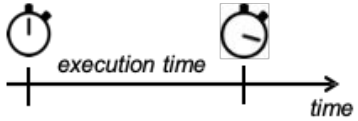
Active Attacks: physically operate on the device to gather informations about secret (e.g. fault injection or microprobing)

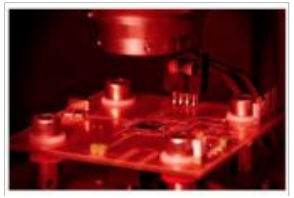
Active vs. passive attacks:

Active attacks exploit side-channel inputs Passive attacks exploit side-channel outputs

What are the Side-Channels Used?

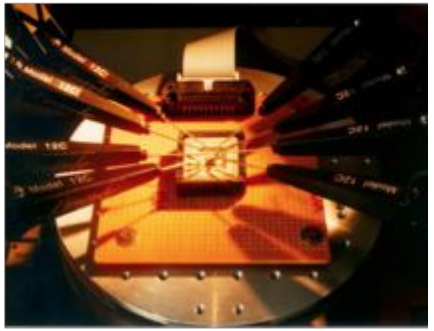
- **Passive**:
  - Timing (Spectre Meltdown)
    - Overall or “local” execution time
  - Power, Electromagnetic (EM) radiation
    - Predominant CMOS technology
    - Dynamic power consumption
    - Electric current induces an EM field
  - More exotic but shown to be practical
    - Sound, temperature, …
- **Invasive**: Photonic emissions

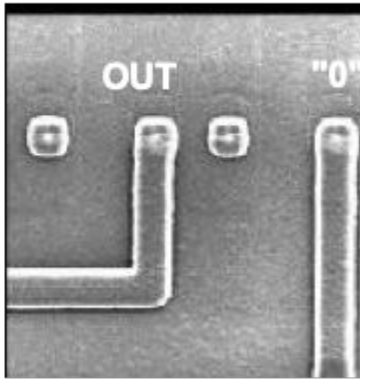




Invasive Attacks

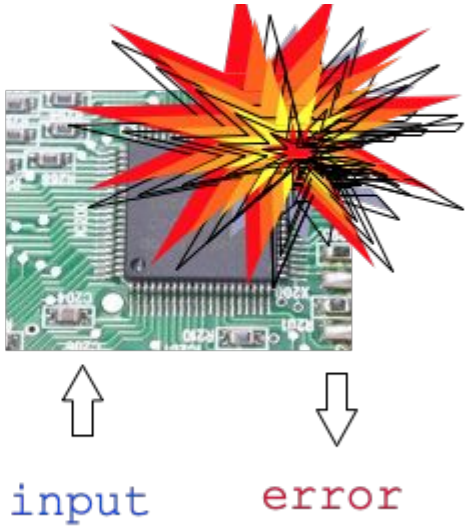
- **<u>Passive:</u> micro-probing**
  - Probe the bus with a very thin needle
  - Read out data from bus or individual cells
- directly
  - Several needles concurrently
- **<u>Active</u>: circuit modification**
  - Connect or disconnect security mechanism
    - Disconnect security sensors
    - RNG stuck at a fixed value
    - Reconstruct blown fuses
  - Cut or paste tracks with laser or focused ion beam
  - Add probe pads on buried layers





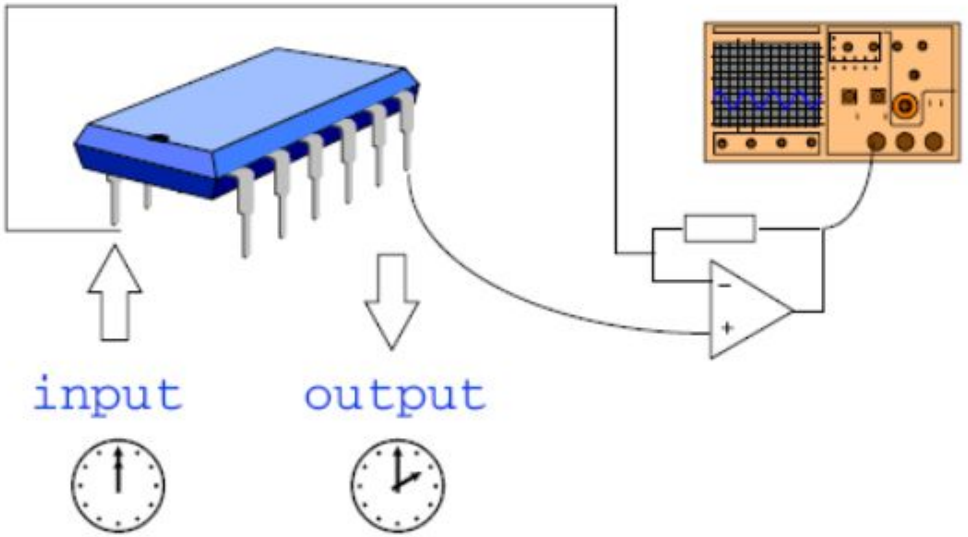
Fault Injection Attacks

- Non-(semi)invasive: apply combination of unaccounted environmental conditions
  - Vcc
  - Glitch
  - Clock
  - Temperature
  - UV
  - Light
  - X-Rays
- And bypass security mechanisms or infer secrets

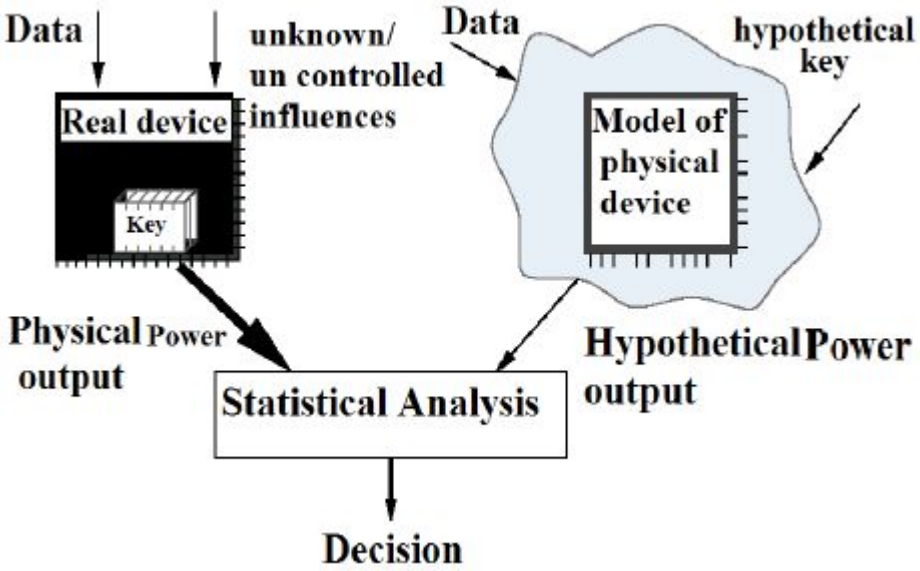


Example: Power Attacks

- Measure the circuit's processing time and current consumption to infer what is going on inside it.



Example: Power Attacks (cont'd)



Correlate the actual power consumption with a model to infer the key

Countermeasures

- **Hiding** -- reduce the SNR by either increasing the noise or reducing the signal
  - Noise Generators, Balanced Logic Styles, Asynchronous Logic, Low Power Design and Shielding
- **Masking/Blinding** -- remove the correlation between the input data and the side-channel emissions from intermediate nodes in the functional block
- **Design Partitioning** -- separate regions of the chip that operate on plaintext from regions that operate on ciphertext

Architectural countermeasures

- What about Timing Attacks?
  - they rely on microarchitectural vulnerabilities that allow the attacker to infer the secret by monitoring execution delays
  - execution based attacks → based on repetition, they show a fingerprint
- Add an online checker to analyze and detect potential malicious activity
  - exploit open ISA such as RISC-V (more later)

Thanks for your attention

Now Ing. Palumbo will introduce some of our current research activities.

more details about this material:

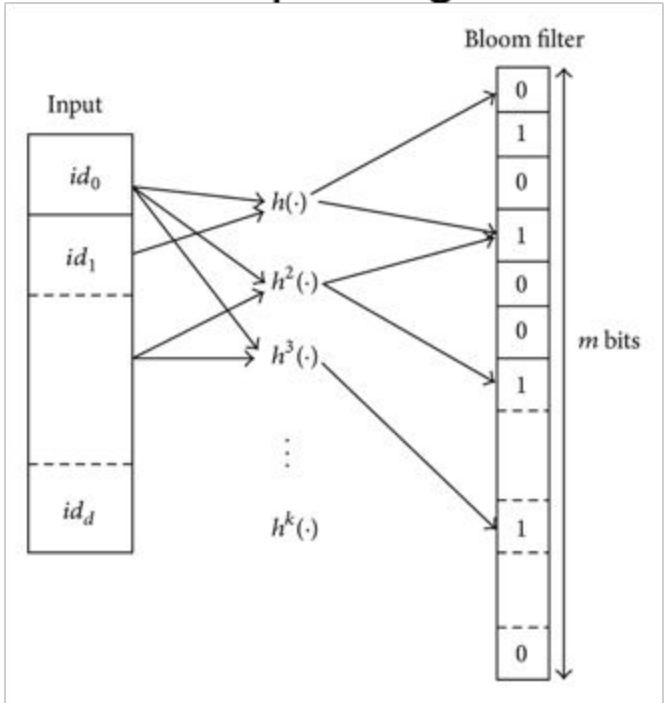
Bhunia, Swarup, and Mark Tehranipoor. ***Hardware security: a hands-on learning approach***. Morgan Kaufmann, 2018.

Our Countermeasures - Example

- Bloom Filter based
  - Probabilistic data structure that is used to test if an element is a member of a set (false positive matches are possible; false negative matches are not possible)
- Count-Min Sketch
  - Counters pointed by the output of hash functions are incremented.
  - This structure will be utilized as a pattern detector.

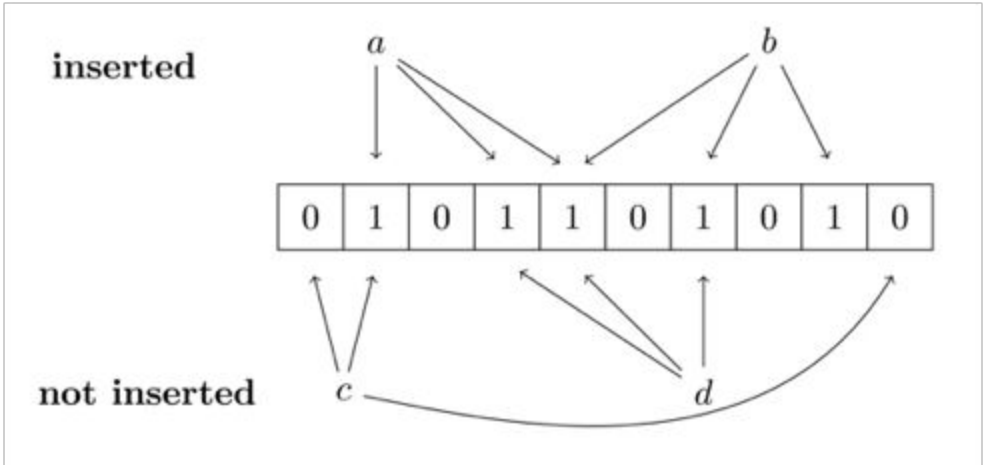
Bloom Filters Overview

- Pushing Process
  - Writing 1 in the corresponding locations of the array(s)



Bloom Filters Overview

- Pulling Process: Reading neatly from the array(s)
  - If we read 1 → the input may be present;
  - If we read 0 → the input for sure is not present in the set



Example Bloom Filter - Algorithm

- H1(x) = x mod 5
- H2(x) = (2x + 3) mod 5

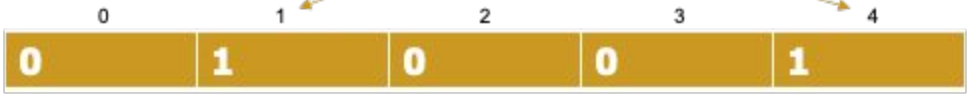
Insert	H1(x)	H2(x)

0	1	2	3	4
0	0	0	0	0

Example Bloom Filter - Algorithm

- H1(x) = x mod 5 → H1(9) = 9 mod 5 = 4
- H2(x) = (2x + 3) mod 5 → H2(9) = (2 * 9 + 3) mod 5 = 1

Insert	H1(x)	H2(x)
9	4	1



Example Bloom Filter - Algorithm

- H1(x) = x mod 5 → H1(11) = 11 mod 5 = 1
- H2(x) = (2x + 3) mod 5 → H2(11) = (2 * 11 + 3) mod 5 = 0

Insert	H1(x)	H2(x)
9	4	1
11	1	0

Example Bloom Filter - Query

- H1(x) = x mod 5 → H1(15) = 15 mod 5 = 0
- H2(x) = (2x + 3) mod 5 → H2(15) = (2 * 15 + 3) mod 5 = 3

Query	H1(x)	H2(x)
15	0	3

- x = 15 isn't present

Example Bloom Filter - Query

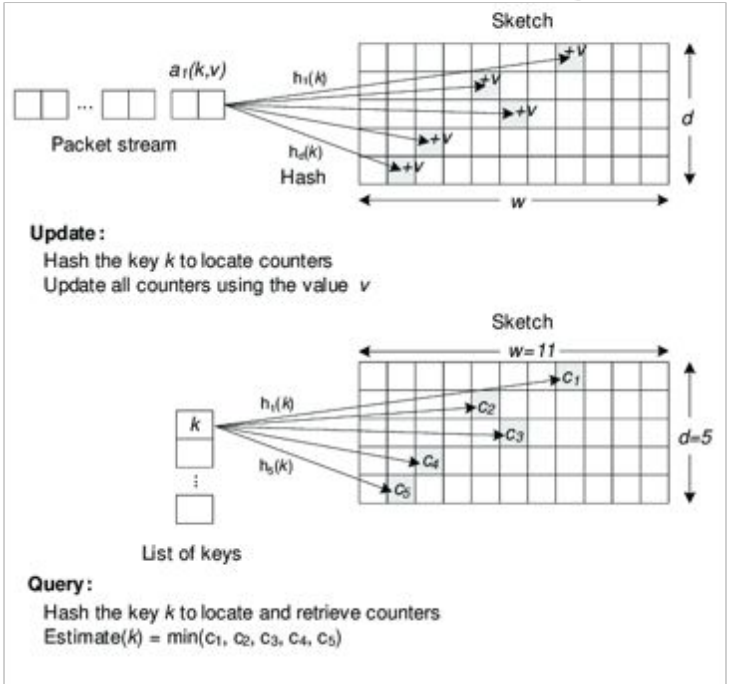
- H1(x) = x mod 5 → H1(16) = 16 mod 5 = 1
- H2(x) = (2x + 3) mod 5 → H2(15) = (2 * 15 + 3) mod 5 = 3

Query	H1(x)	H2(x)
15	0	3
16	1	0

- x = 16 may be present

Count-Min Sketch Overview

- Counters pointed by the output of hash functions are incremented.
  - This structure will be utilized as a pattern detector.



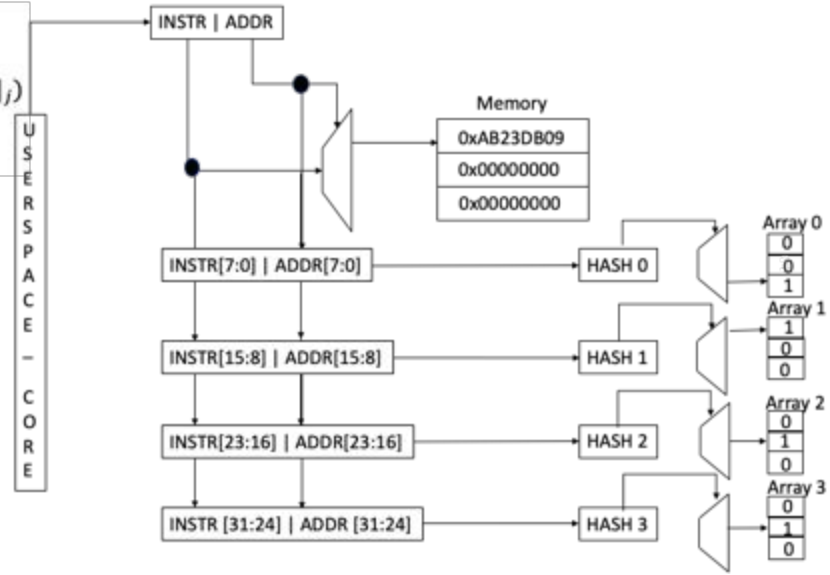
Hardware Trojan Checker

- Pushing data (Golden Run)

$$Array_i[Address_j] = 1$$

$$Address_j = Hash_i(Instr[m:n]_j | AddrOfInstr[m:n]_j)$$

$$Hash_i(Input) = [Input] mod 2^{16}$$



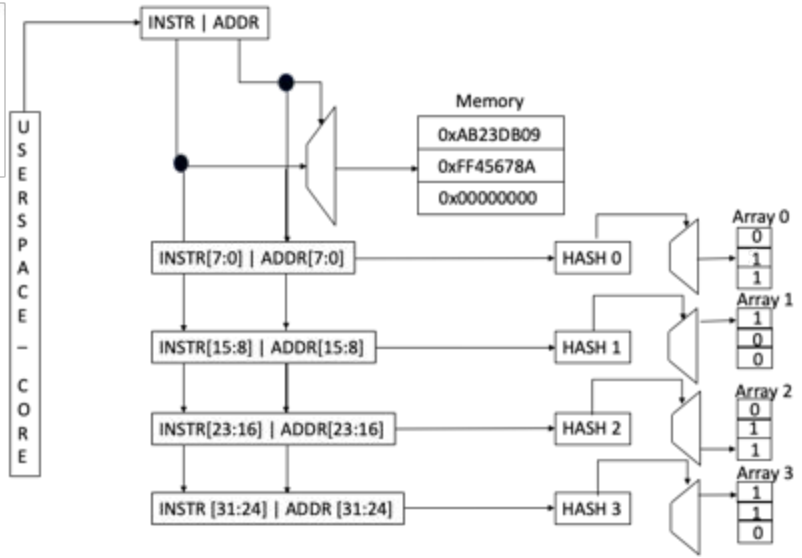
Hardware Trojan Checker

- Pushing data (Golden Run)

$$Array_i[Address_j] = 1$$

$$Address_j = Hash_i(Instr[m:n]_j | AddrOfInstr[m:n]_j)$$

$$Hash_i(Input) = [Input] mod 2^{16}$$



Hardware Trojan Checker

- Pulling data (possibly)

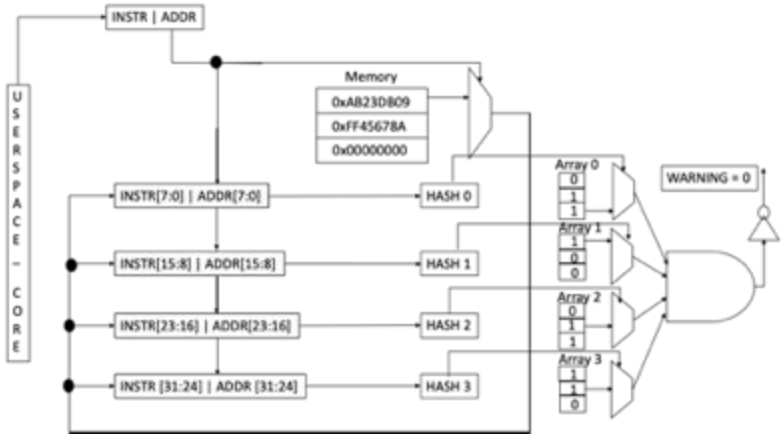
$$BitOutArray_i[Address_j] = Array_i[Address_j]$$

$$Address_j = Hash_i(DataMemory[m:n]_j | AddrOfInstr[m:n]_j)$$

$$Warning = \prod \overline{BitOutArray_i}$$

$$Hash_i(Input) = [Input] mod 2^{16}$$

$$AddrOfInstr = AddrOfDataMemory$$



Hardware Trojan Checker

- Pulling data with trojan

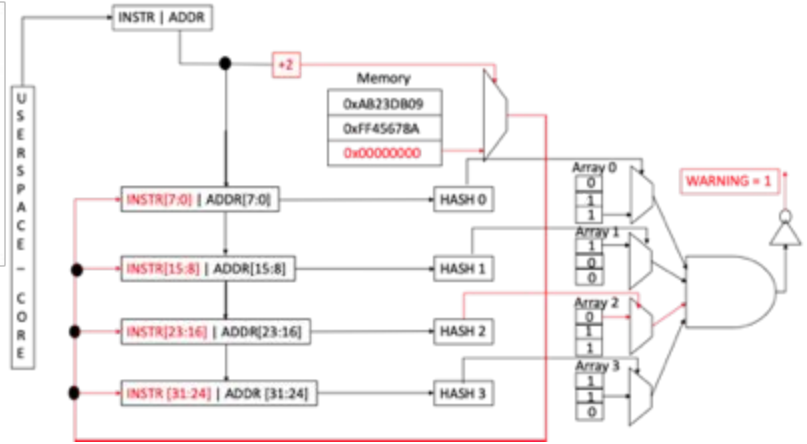
$$BitOutArray_i[Address_j + T] = Array_i[Address_j]$$

$$Address_j = Hash_i(DataMemory[m:n]_j | AddrOfInstr[m:n]_j)$$

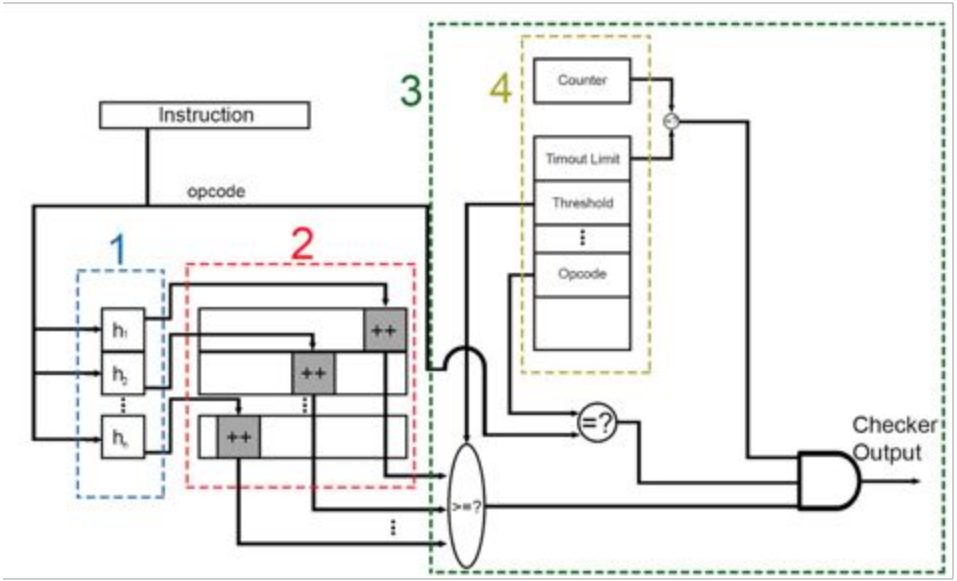
$$Warning = \prod \widetilde{BitOutArray_i}$$

$$Hash_i(Input) = [Input] mod 2^{16}$$

$$AddrOfInstr \neq AddrOfDataMemory$$

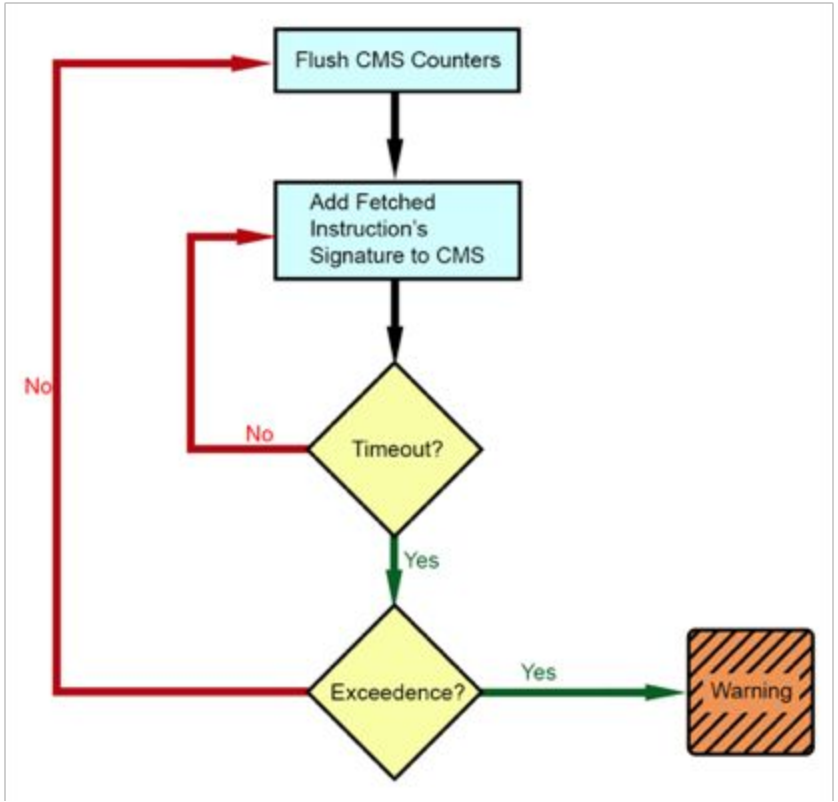


SCA Checker



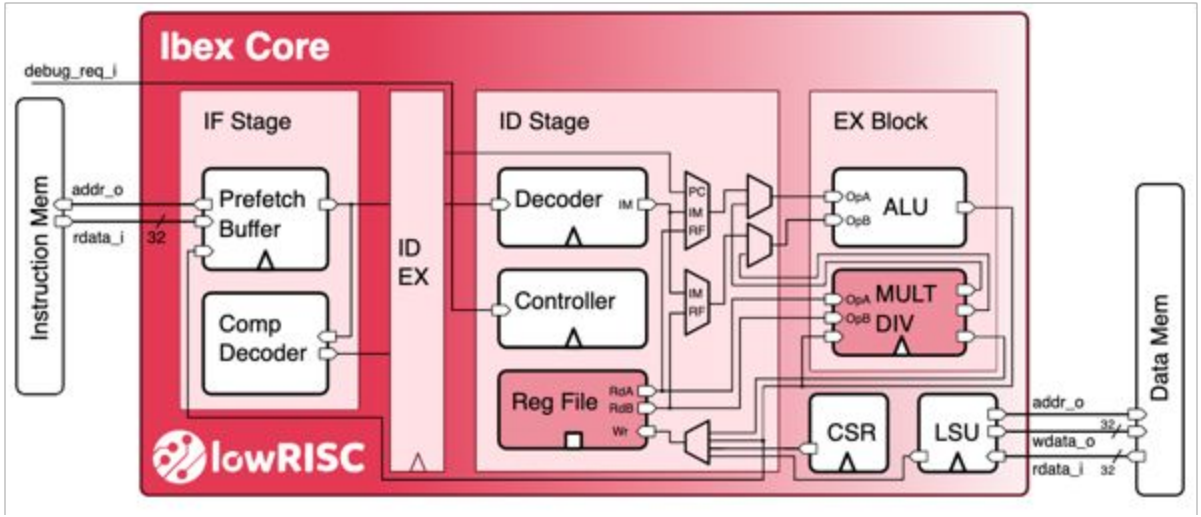
(1): Hash Logic (HL)
(2): CMS Memory (CMSM)
(3): Comparison Machine (CM)
(4): Model Description Register Unit (MDRU)

SCA Checker - Workflow



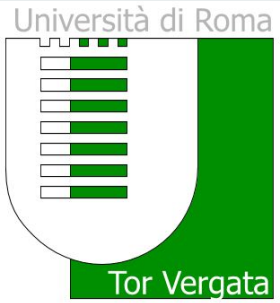
Our Methodology

- Platform: Ibex Core (32-bit, in-order, 2-stage pipeline, RV32IMCISA)
- FPGA implementation



Future & Ongoing Work

- Our systems can be emulated on a multi-core architecture
- Detection and implementation of architecture to mitigate reveal attacks to the clock of the core
- Golden VS BlackBox
  - To catalog: study features of the circuit in order to associate those characteristics with the presence of a specific type of trojan (temperature, power, EM emission, throughput, area, etc.)



Introduction to Hardware Security

Prof. Marco Ottavi

Ing. Alessandro Palumbo

Thanks. Q&A?