POLITECNICO MILANO 1863

Advanced topics in Hardware Security
-
Hardware Trojan Horses

Luca Cassano
luca.cassano@polimi.it
cassano.faculty.polimi.it/HW_sec.html

Acknowledgment

Prof. Mark Tehranipoor

University of Florida (US)

Prof. Marco Ottavi

University of Rome Tor Vergata (IT) and University of Twente (NL)

Course calendar

Date			Topic
March	6	Mon	Course introduction – perspective
March	10	Fri	Students presentations
March	13	Mon	Hardware Trojans Horses
March	17	Fri	Students presentations
March	20	Mon	Microarchitectural SCAs
March	24	Fri	Students presentations
March	27	Mon	IP Piracy
March	31	Fri	Students presentations
April	3	Mon	Logic locking
April	6	Thu	Students presentations

Luca Cassano

Christian Pilato

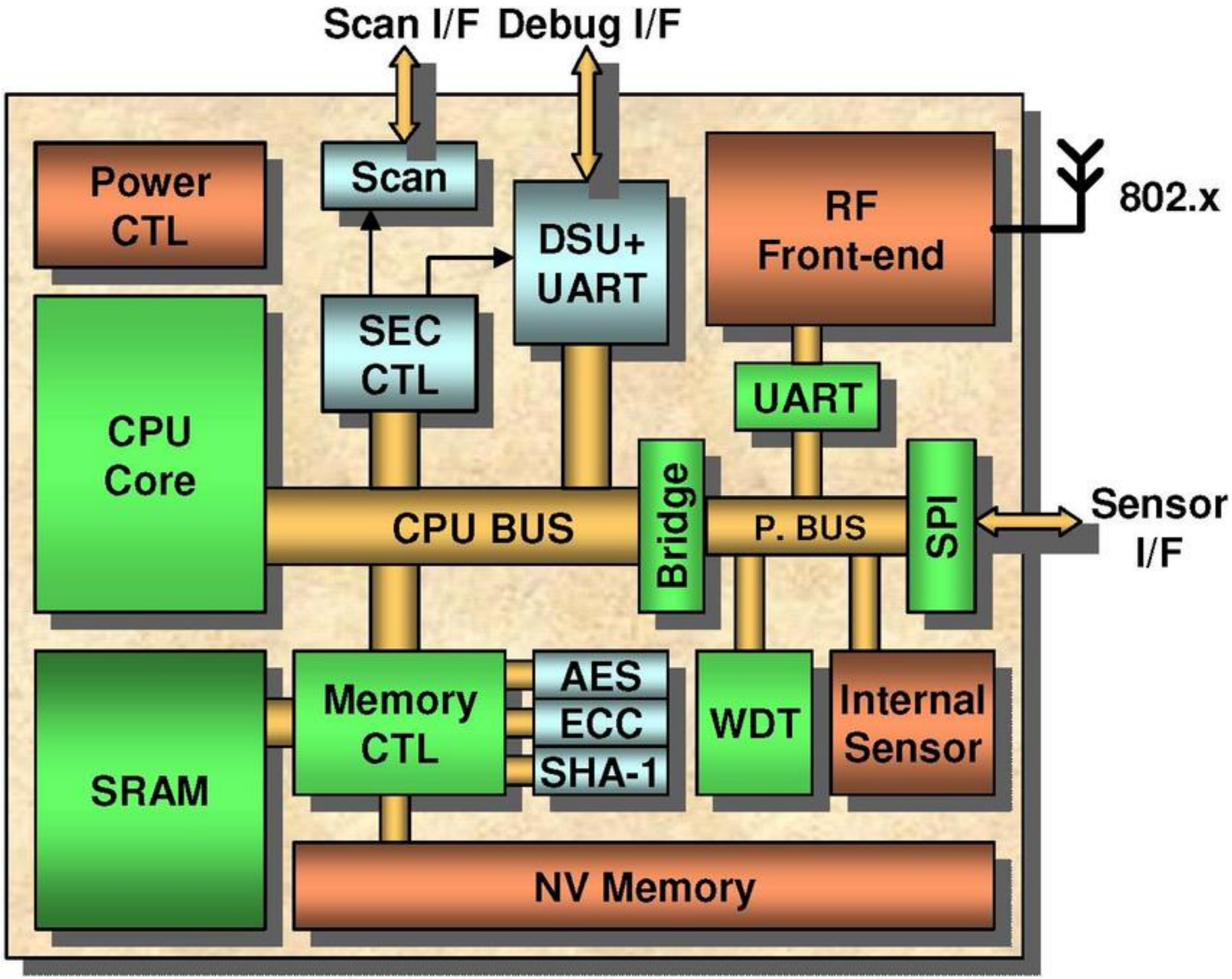
Monday, 9:15 – 12:15, room 3.1.8

Friday, 9:15 – 11:15, room 2.2.3

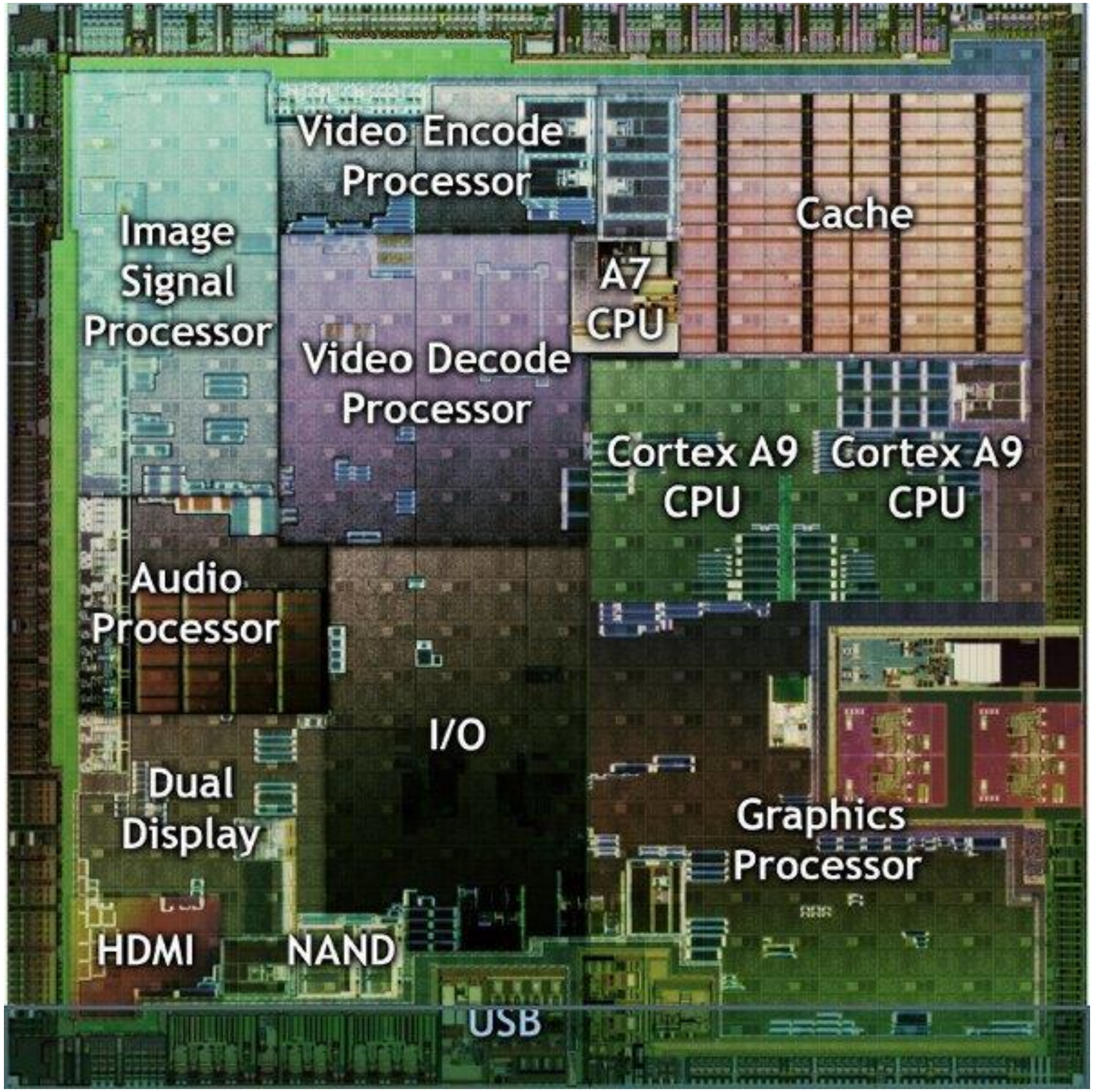
Thursday, April 6th 9:15 – 11:15, room 25.1.4

Recap: where hardware security issues come from?

The System on Chip Model



The System on Chip Model



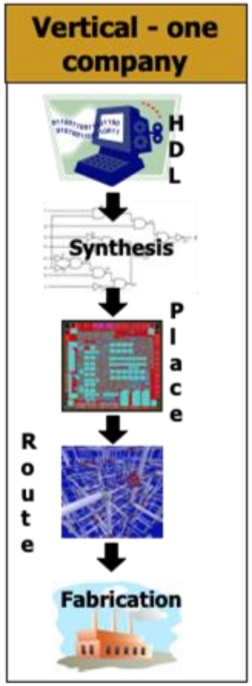
VLSI Industry: Business Model trends

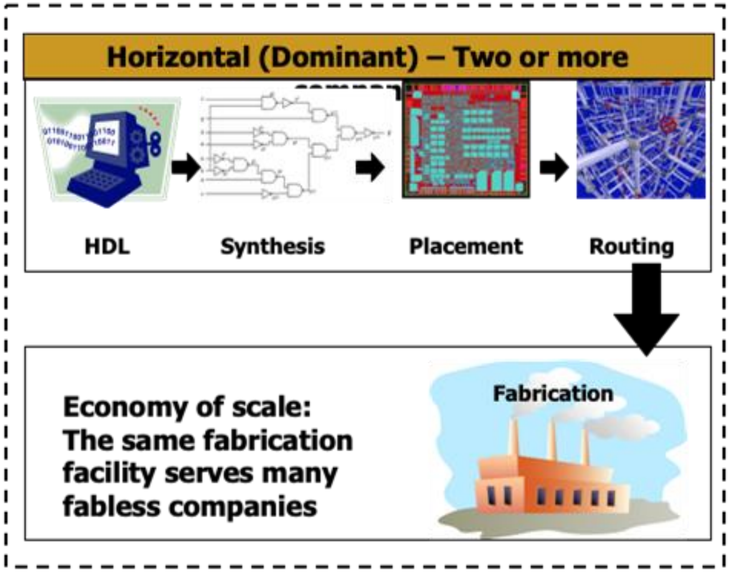
Vertical Model:

all in-house development, high costs, low economy of scale

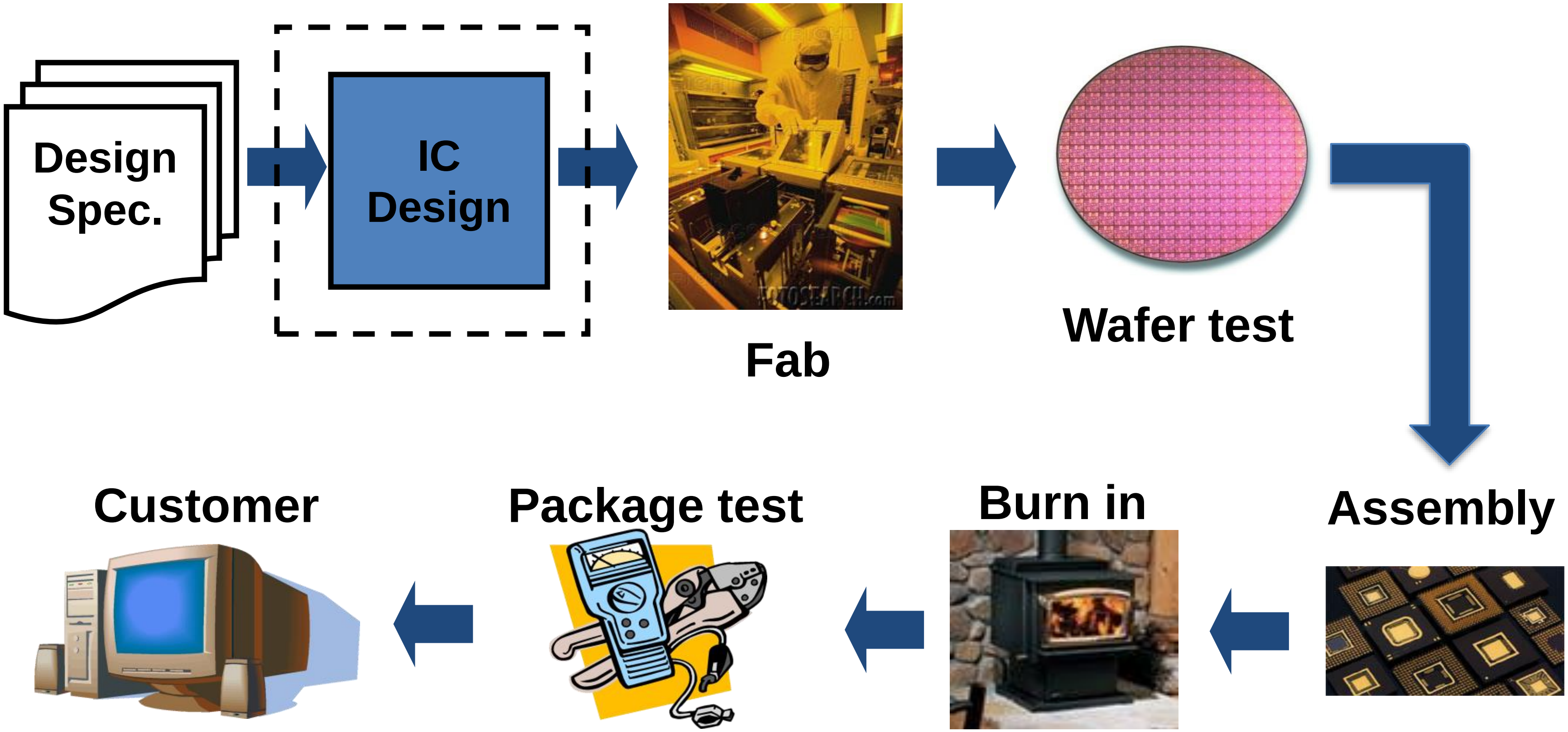
Horizontal Model:

several companies involved, lower costs, economy of scale

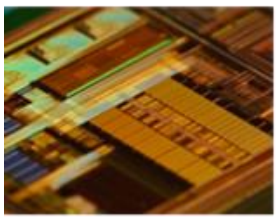


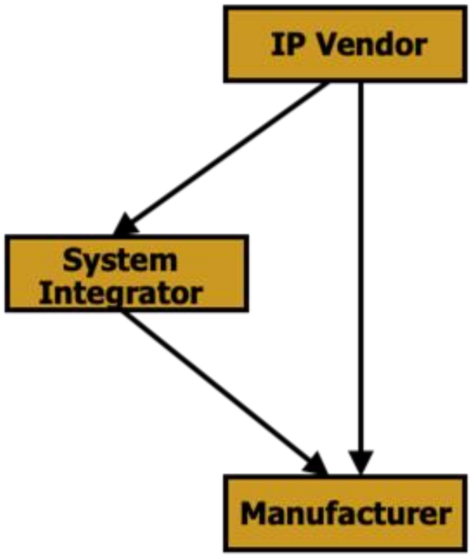


IC Design and Test Flow



Can we Trust Hardware?





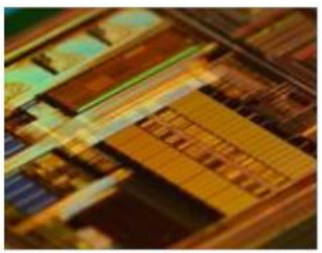
Any of these steps can be untrusted

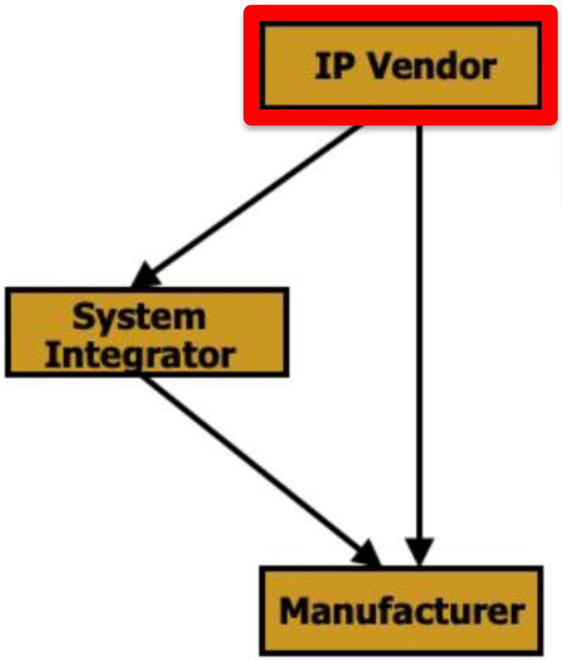
IP: Intellectual properties sometimes **provided by third party vendors**

System Integrator **combines several IPs** into a chip design

Manufacturer fabricates the chips **based on the received design**

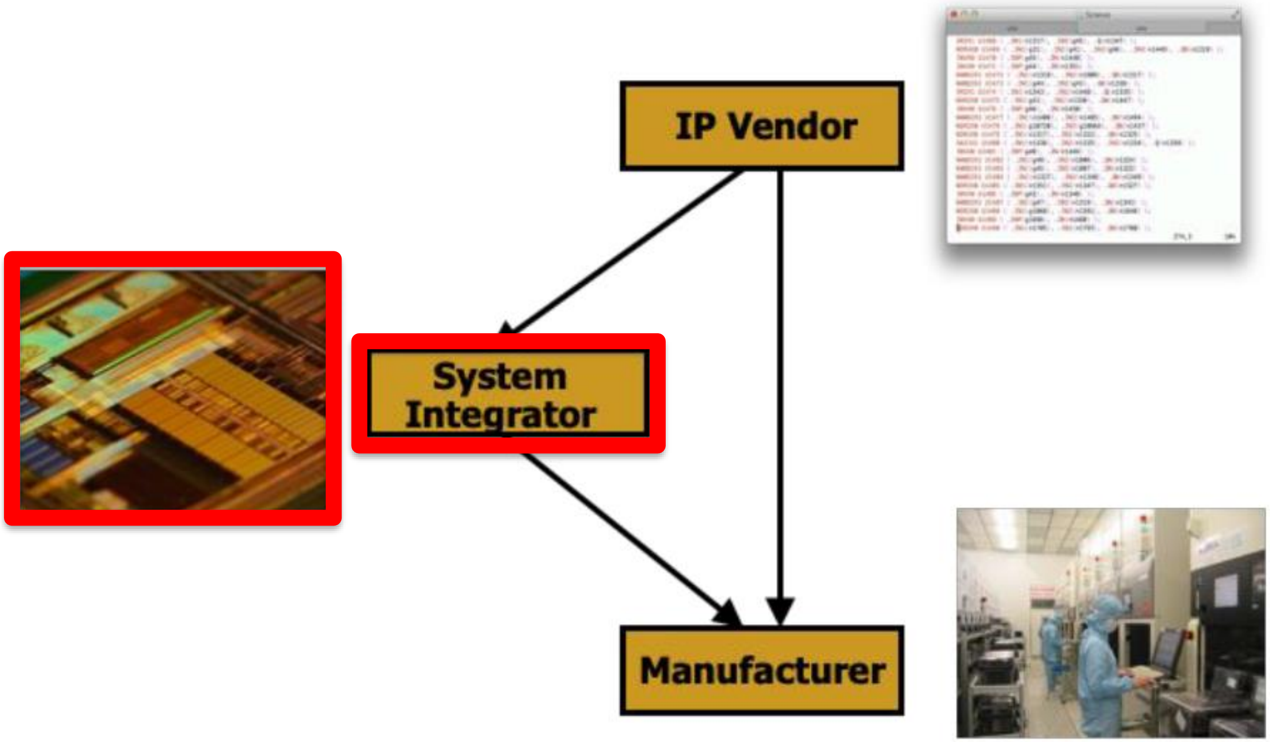
Can we Trust Hardware?





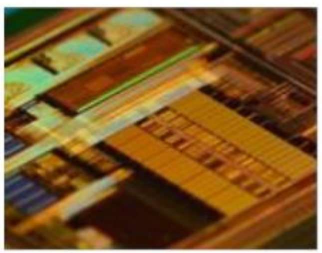
- HW Trojan Horses

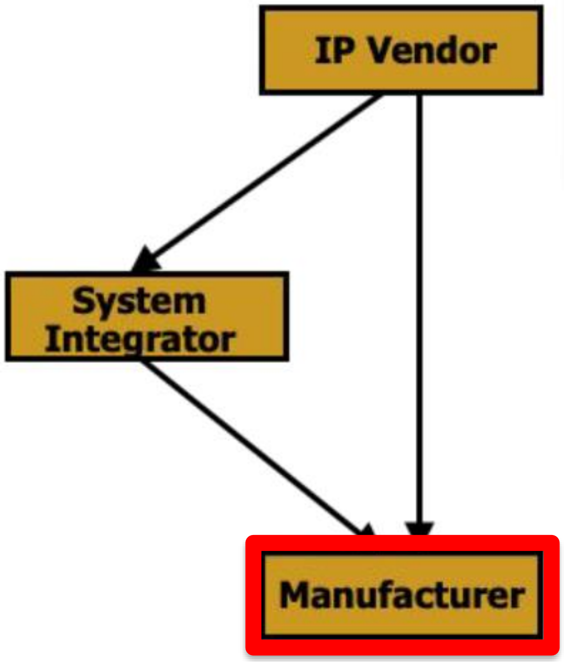
Can we Trust Hardware?

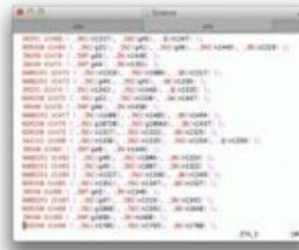


- HW Trojan Horses
- IP theft
- Malicious CAD tools

Can we Trust Hardware?







- HW Trojan Horses
- IP theft
- Off spec. and Defective ICs
- Overproduced ICs
- Recycled ICs

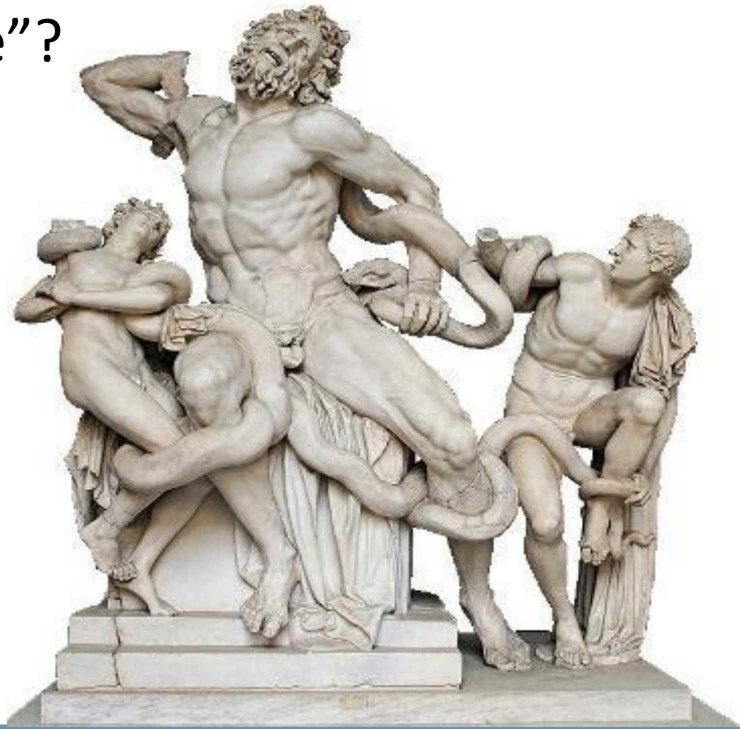
Hardware Trojan Horses
-
Motivations and Examples

Hardware Trojan Horses

timeo Danaos et dona ferentes
Virgilius (Aeneid II, 49)

Hardware Trojan Horses

- The first case of “hardware trust issue” in history
- Can I trust what is inside of a “hardware”?
- Laocoön was right, **trust but check**



Hardware Trojan Horses

What a Hardware Trojan is:

- A **malicious addition or modification** to the existing circuit elements.

What can hardware Trojans do?

- Change the functionality
- Reduce the reliability (up to denying the service)
- Leak valuable information

Hardware Trojan Horses

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
- …

Hardware Trojan Horses

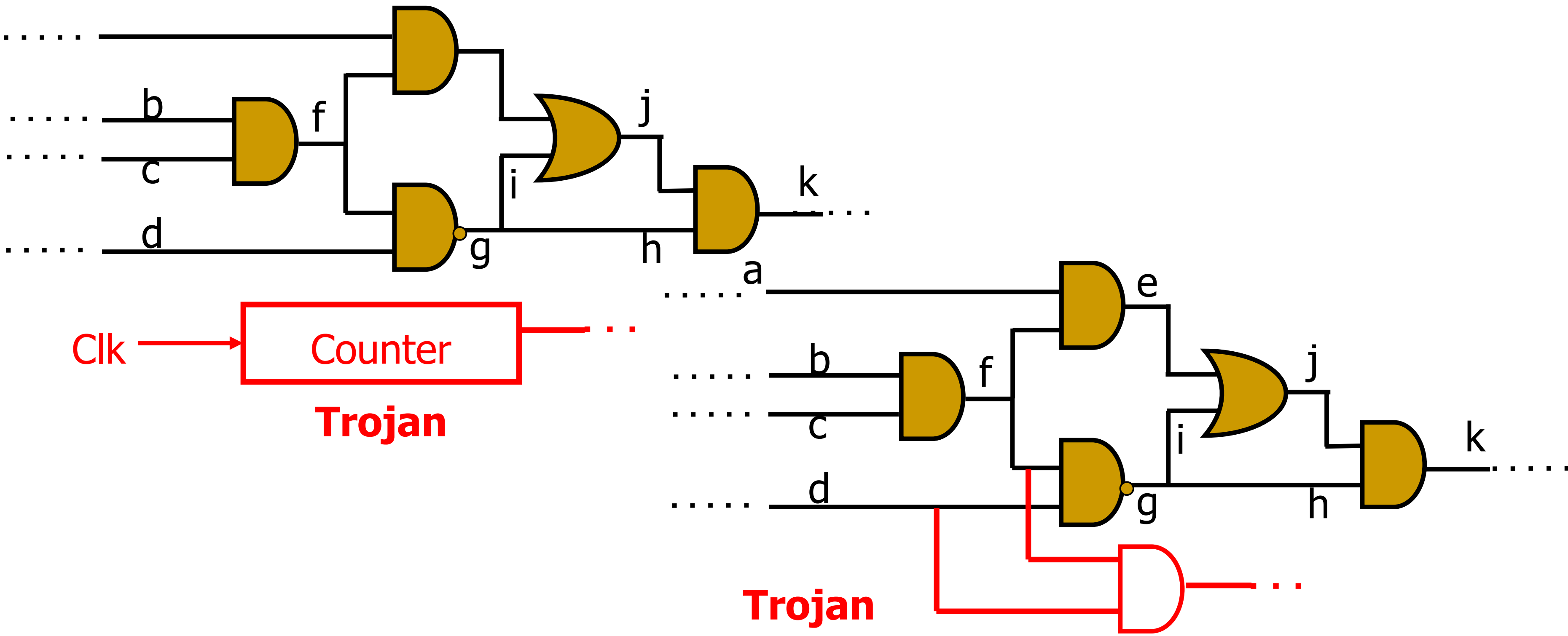
A hardware Trojan horse is generally composed of a:

- A **triggering mechanism** (what activates the HTH)
- A **payload** (what the HTH makes after being activated)

Hardware Trojan Horses

A hardware Trojan horse is generally composed of a:

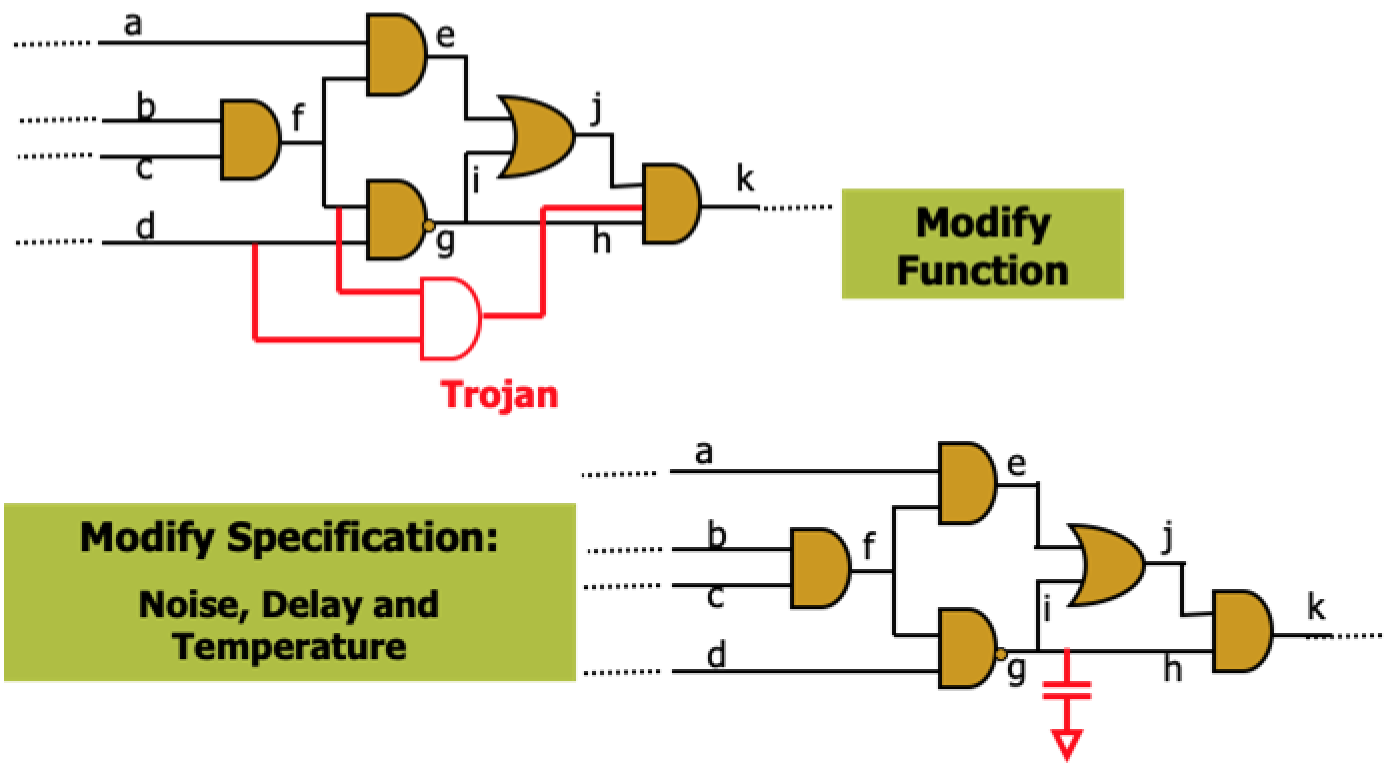
- A **triggering mechanism** (what activates the HTH)



Hardware Trojan Horses

A hardware Trojan horse is generally composed of a:

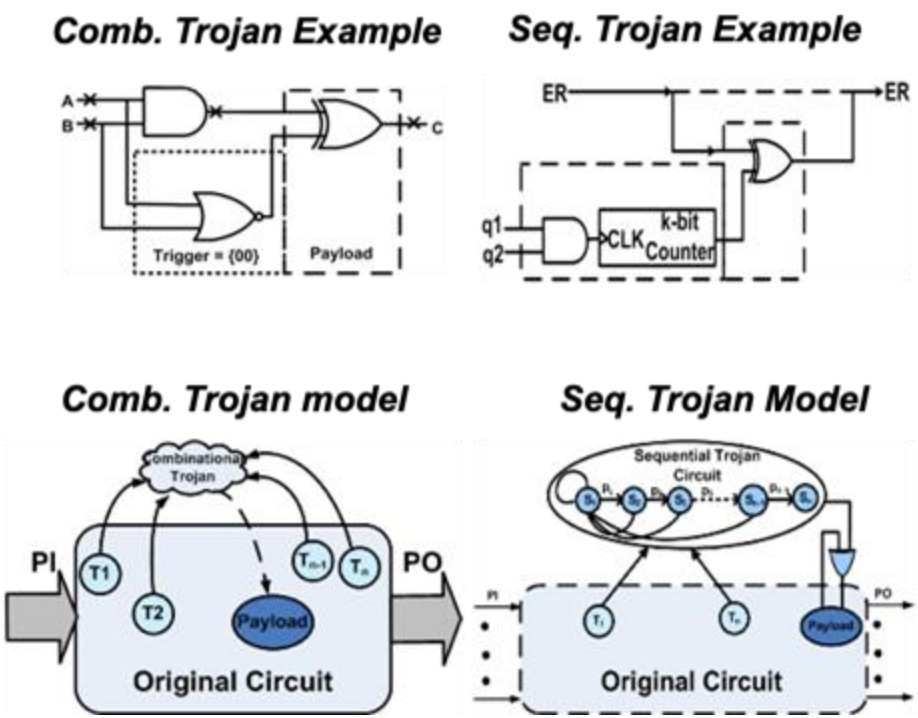
- A **payload** (what the HTH makes after being activated)



Hardware Trojan Horses

A hardware Trojan horse is generally composed of a:

- A **triggering mechanism** (what activates the HTH)
- A **payload** (what the HTH makes after being activated)



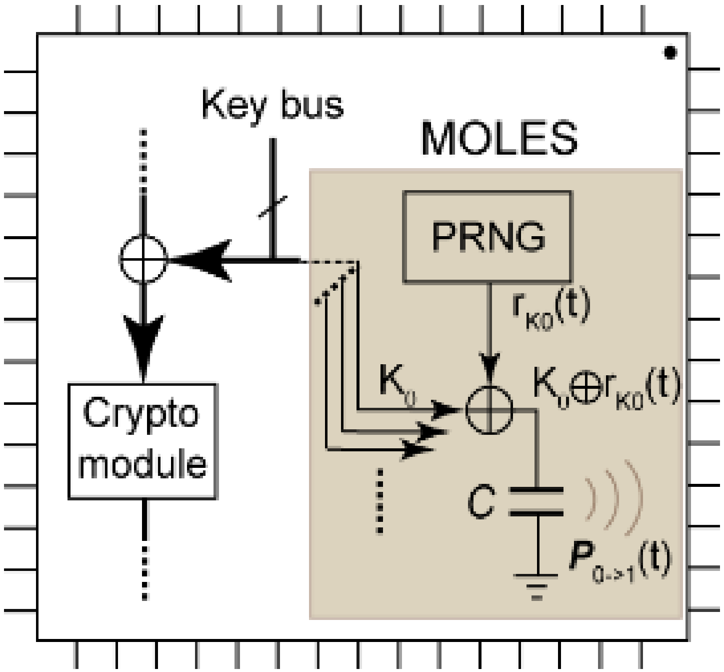
Hardware Trojan Horses

The **Outside the Box** Israeli Air Force operation:

- Eight fighter planes from IAF attacked and destroyed a nuclear plant (under construction) in Deir ez-Zor, Syria, in 2007
- All Syrian radars and air defense missile systems switched-off simultaneously
- Syrian defense equipments featured COTS components (probably manufactured by Intel Israel)
- All these information have been confirmed by IAF in 2018

Hardware Trojan Horses

MOLES*: Info Leakage Trojan

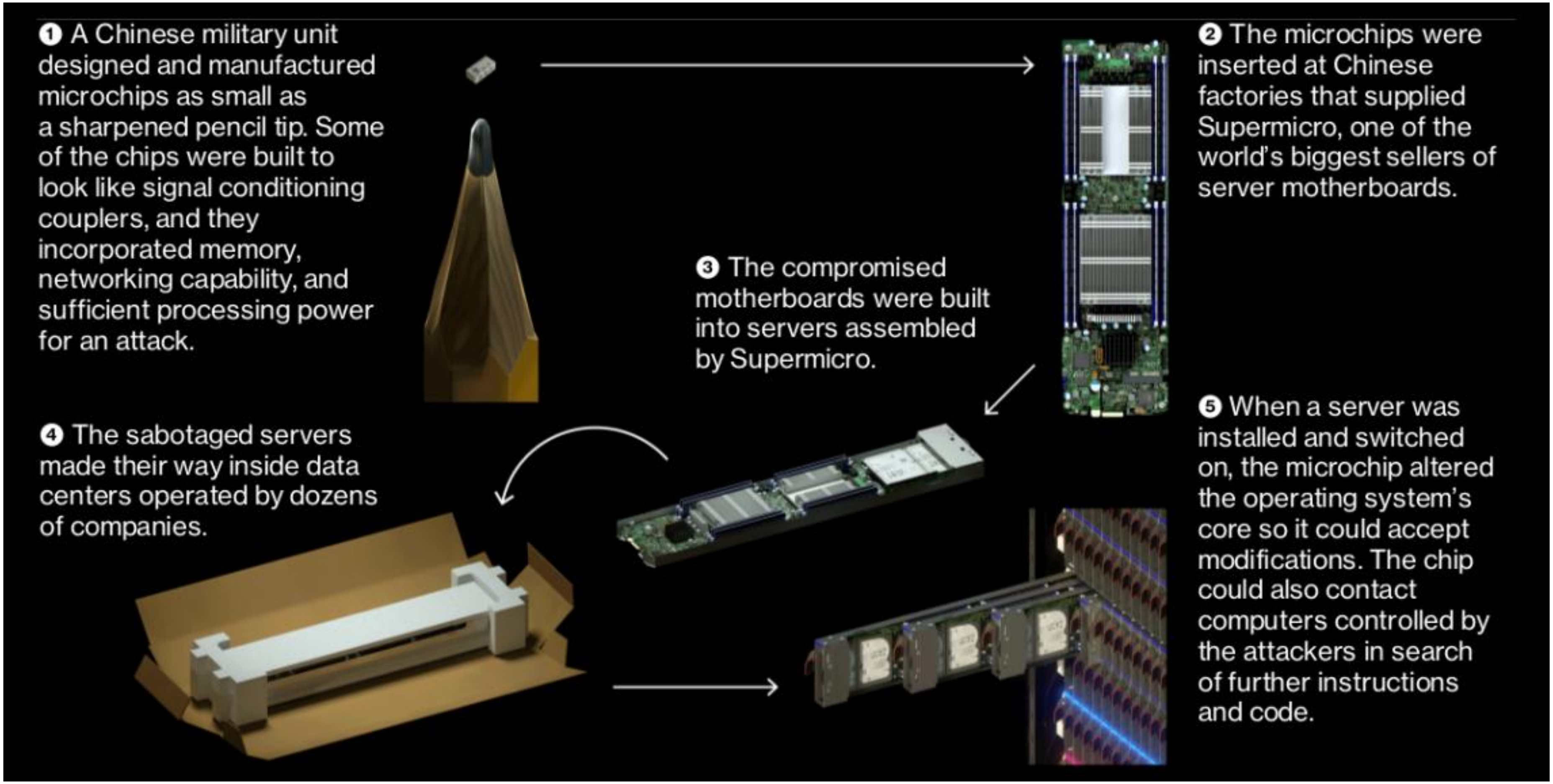


Malicious Off-chip Leakage Enabled by Side-channels

- Leak the secret key by using an on chip modification that creates a power consumption pattern related to the values of the key.
- The PNRG obfuscates the leakage

L. Lin, W. Burleson and C. Paar, "MOLES: Malicious off-chip leakage enabled by side-channels," *2009 IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers*, San Jose, CA, 2009, pp. 117-122.

Hardware Trojan Horses



www.bloomberg.com

Hardware Trojan Horses

SPIEGEL International

Catalog Advertises NSA Toolbox

After years of speculation that electronics can be accessed by intelligence agencies through a back door, an internal NSA catalog reveals that such methods already exist for numerous end-user devices.

Von Jacob Appelbaum, **Judith Horchert** und **Christian Stöcker**
29.12.2013, 09.19 Uhr

https://www.spiegel.de/international/world/catalog-reveals-nsa-has-back-doors-for-numerous-devices-a-940994.html

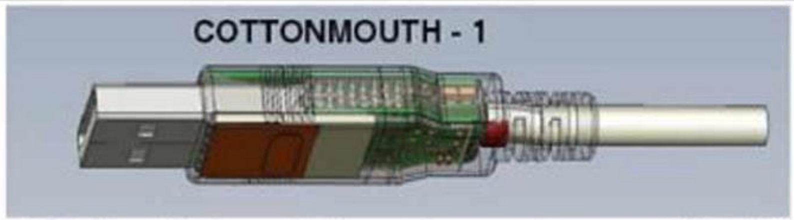
Hardware Trojan Horses

COTTONMOUTH-I

ANT Product Data

08/05/08

(TS//SI//REL) COTTONMOUTH-I (CM-I) is a Universal Serial Bus (USB) hardware implant which will provide a wireless bridge into a target network as well as the ability to load exploit software onto target PCs.



(TS//SI//REL) CM-I will provide air-gap bridging, software persistence capability, "in-field" re-programmability, and covert communications with a host software implant over the USB. The RF link will enable command and data infiltration and exfiltration. CM-I will also communicate with Data Network Technologies (DNT) software (STRAITBIZARRE) through a covert channel implemented on the USB, using this communication channel to pass commands and data between hardware and software implants. CM-I will be a GENIE-compliant implant based on CHIMNEYPOOL.

(TS//SI//REL) CM-I conceals digital components (TRINITY), USB 1.1 FS hub, switches, and HOWLERMONKEY (HM) RF Transceiver within the USB Series-A cable connector. MOCCASIN is the version permanently connected to a USB keyboard. Another version can be made with an unmodified USB connector at the other end. CM-I has the ability to communicate to other CM devices over the RF link using an over-the-air protocol called SPECULATION.

COTTONMOUTH CONOP
INTERNET Scenario

https://www.eff.org/files/2014/01/06/20131230-appelbaum-nsa_ant_catalog.pdf

Hardware Trojan Horses

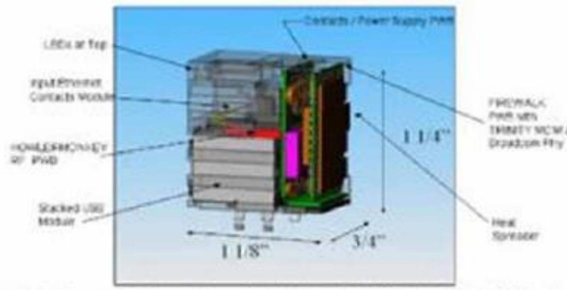
TOP SECRET//COMINT//REL FVEY

FIREWALK

ANT Product Data

(TS//SI//REL) FIREWALK is a bidirectional network implant, capable of passively collecting Gigabit Ethernet network traffic, and actively injecting Ethernet packets onto the same target network.

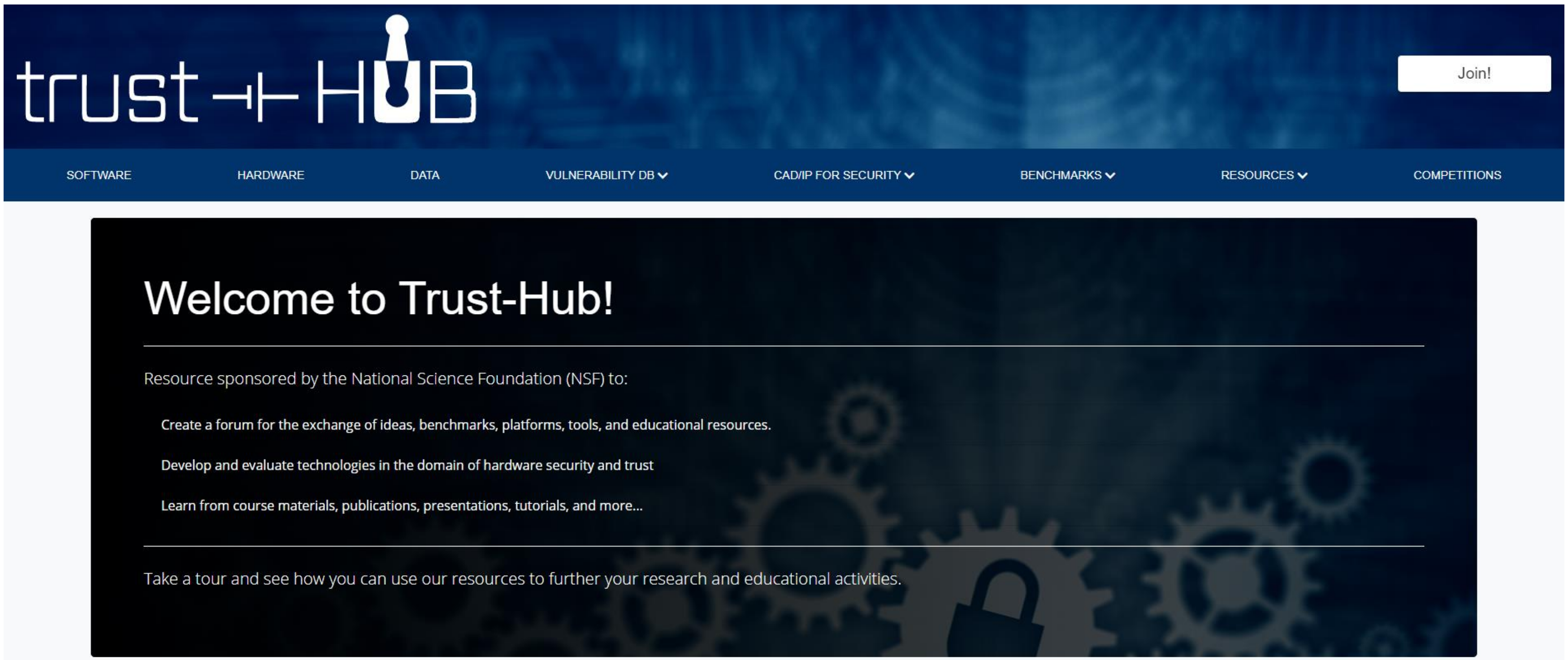
08/05/08



(TS//SI//REL) FIREWALK is a bi-directional 10/100/1000bT (Gigabit) Ethernet network implant residing within a dual stacked RJ45 / USB connector. FIREWALK is capable of filtering and egressing network traffic over a custom RF link and injecting traffic as commanded; this allows a ethernet tunnel (VPN) to be created between target network and the ROC (or an intermediate redirector node such as DNT's DANDERSPRITZ tool.) FIREWALK allows active exploitation of a target network with a firewall or air gap protection.
(TS//SI//REL) FIREWALK uses the HOWLERMONKEY transceiver for back-end communications. It can communicate with an LP or other compatible HOWLERMONKEY based ANT products to increase RF range through multiple hops.

https://www.eff.org/files/2014/01/06/20131230-appelbaum-nsa_ant_catalog.pdf

Hardware Trojan Horses - *trust-hub.org*



Hardware Trojan Horses - *trust-hub.org*

BASICRSA-T400

time-based | design | register-transfer | denial-of-service | processor

Trojan leaks inExp (private key exponent (e)), and after a certain number of encryption Trojan replaces the secret key to deny the service. The adversary would be the only entity would understand the message.

Contributed By

Hassan Salmani

University of Connecticut

Resources

File 1

Hardware Trojan Horses - *trust-hub.org*

MC8051-T400

time-based | design | register-transfer | functional | processor | change-functionality

The Trojan is triggered when a specific sequence of commands is executed. The Trojan disables handling interrupt after activation.

Contributed By

Hassan Salmani

University of Connecticut

Resources

File 1

Hardware Trojan Horses - *trust-hub.org*

MC8051-T800

design | user-input | register-transfer | functional | denial-of-service | processor

The Trojan manipulates the stack pointer when specific data are received through UART.

Contributed By

Hassan Salmani

University of Connecticut

Resources

File 1

Hardware Trojan Horses - *trust-hub.org*

PIC16F84-T200

time-based | design | register-transfer | functional | denial-of-service | processor

The Trojan trigger, a state machine, observes the number of execution of specific instruction. Above a certain number of execution the Trojan is triggered, and it replaces the instruction register with sleep command.

Contributed By

Hassan Salmani

University of Connecticut

Resources

File 1

Hardware Trojan Horses

In the past HTHs have been considered an academical issue due to:

- **Difficulty to be inserted** at the layout-level
- **Reduced complexity and dangerousness**

Hardware Trojan Horses

The large use of third-party IP cores makes systems more exposed to HTHs insertion and more difficult to be verified

Moreover, the novel menace of **Software-Exploitable HTHs** raised

- By running specific assembly sequences the authors:
  - Leaked the entire program
  - Forced unexpected memory read/write
  - Modified the content of the program counter and the stack pointer
  - Modified the input values of the ALU

- X. Wang, T. Mal-Sarkar, A. Krishna, S. Narasimhan and S. Bhunia, "Software exploitable hardware Trojans in embedded processor," 2012 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Austin, TX, USA, 2012, pp. 55-58, doi: 10.1109/DFT.2012.6378199.

Hardware Trojan Horses

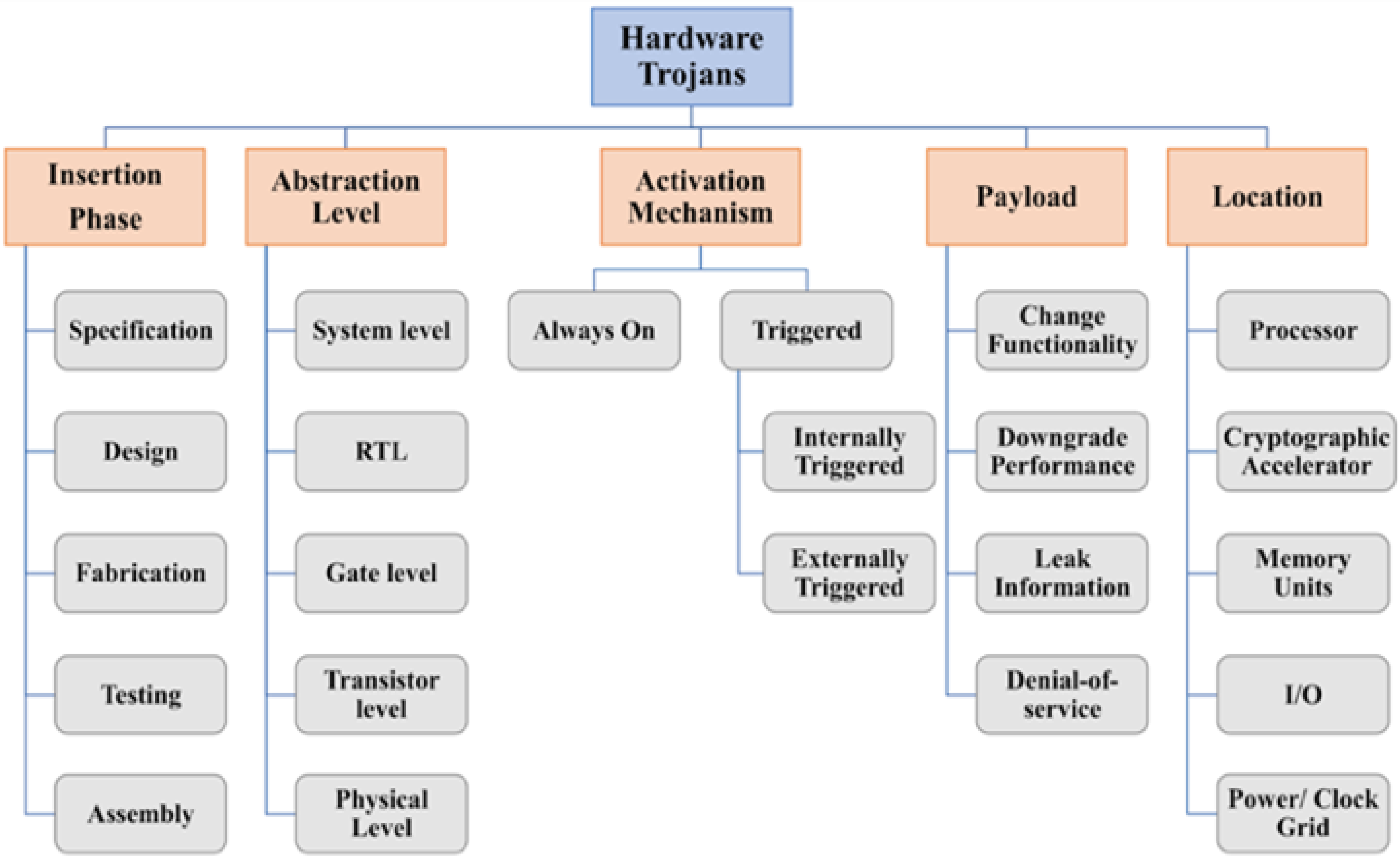
An *"undocumented feature"* that allows an unauthorized privilege escalation through the execution of `.byte 0x0f, 0x3f` has been found in the Via Technologies C3 processor

The C-series processors had been marketed in mid 2000s towards industrial automation, point-of-sale, ATM, and healthcare hardware, as well as a variety of consumer desktop and laptop computers.

- C. Domas, Hardware backdoors in x86 cpus, 2018, https://i.blackhat.com/us-18/Thu-August-9/us-18-Domas-God-Mode-Unlocked-Hardware-Backdoors-In-x86-CPUs-wp.pdf
- Project:rosenbridge, 2022, URL https://github.com/xoreaxeaxeax/rosenbridge

Hardware Trojan Horses
-
Taxonomy

Hardware Trojan Horses



Hardware Trojan Horses: Insertion Phase

Specification Phase: In this phase, functional specifications or other design constraints can be altered.

- Example, a Trojan at the specification phase might change the hardware’s timing requirements.

Hardware Trojan Horses: Insertion Phase

Design Phase: In this phase developers can use third-party IP blocks and standard cells.

- Example: Trojans might be in any of the components that aid the design. For example, a standard cell library can be tampered with Trojans.

Hardware Trojan Horses: Insertion Phase

Fabrication Phase: In this phase, developers create a mask set and use wafers to produce the dies. Examples of Trojans:

- Subtle mask changes can have serious effects. In an extreme case, an adversary can substitute a different mask set.
- Chemical compositions might be altered during fabrication to increase the electromigration in critical circuitry, such as power supplies and clock grids, which would accelerate failures.

Hardware Trojan Horses: Insertion Phase

Testing Phase: The testing phase could be used to insert trojans (for example information leakage through modified scan chains) , and also to detect Trojans, but Testing is only useful for detection when done in a trustworthy manner.

- For example, an adversary who inserted a Trojan in the fabrication phase would want to have control over the test vectors to ensure that the Trojan is not detected during test

Hardware Trojan Horses: Insertion Phase

Assembly Phase: In this phase developers assemble the tested chip and other hardware components on a printed circuit board (PCB). Even if all the ICs in a system are trustworthy, malicious assembly can introduce security flaws in the system.

- For example, an unshielded wire connected to a node on the PCB can introduce unintended electromagnetic coupling between the signal on the board and its electromagnetic surroundings. An adversary can exploit this for information leakage and fault injection.

Hardware Trojan Horses: Abstraction Level

- **System Level:** At the system level, the different hardware modules, interconnections, and communication protocols used are defined. At this level, the Trojans may be triggered by the modules in the target hardware.
  - Example: the ASCII values of the inputs from the keyboard can be interchanged.

Hardware Trojan Horses: Abstraction Level

- **Register-Transfer Level:** At the RTL, chip designers describe each functional module in terms of registers, signals, and Boolean functions. A Trojan can be easily designed and inserted at the RTL because the attacker has full control over the design's functionality.
    - Example: a Trojan implemented at this level might halve the rounds of a cryptographic algorithm by making a round counter to advance in two steps instead of one.

Hardware Trojan Horses: Abstraction Level

- **Gate Level:** At this level, an SoC is represented as an interconnection of logic gates. An attacker can carefully control all aspects of the inserted Trojan, including its size and location.
  - Example: a Trojan might be a simple comparator consisting of basic gates (AND, OR, XOR gates) that monitor the chip's internal signals.

Hardware Trojan Horses: Abstraction Level

- **Transistor Level:** This level gives the Trojan designer control over circuit characteristics, such as power and timing. The attacker can insert or remove individual transistors,,or modify transistor sizes
    - Example: a transistor-level Trojan might be a transistor with low gate width that can cause more delay in the critical path.

Hardware Trojan Horses: Abstraction Level

- **Physical Level:** An attacker can insert Trojans by modifying the size of the wires and distances between circuit elements and reassigning metal layers.
  - Example, changing the width of the clock wires, timing critical nets or metal wires in the chip can cause clock skew.

Hardware Trojan Horses: Activation Mechanism

- **Always On:** these Trojans are always active and do not need a Trigger, their effect may show up in certain specific operating conditions
  - Example: a trojan adding delay on a critical path may cause timing failure in some frequency scaling conditions.

Hardware Trojan Horses: Activation Mechanism

- **Internally Triggered:** activated by an event within the target device. The event might be either time-based or physical-condition-based.
  - Example: Time Bomb triggered by a counter in the design at a predetermined time, or trigger can be when the chip temperature exceeds a certain threshold.

Hardware Trojan Horses: Activation Mechanism

- **Externally Triggered:** requires external input to the target module to activate. The external trigger can be user input or component output. User-input triggers include push buttons, switches, keyboards, or keywords and phrases in the input data stream in a system. Component-output triggers might be from any of the components that interact with the target device.
  - Example, a Trojan in a cryptomodule can derive its trigger condition from the applied plaintext input and triggers when it observes a specific plaintext, or a combination of plaintext and operating conditions.

Hardware Trojan Horses: Payload

- **Change Functionality:** Trojan can change the functionality of the target device, and cause subtle errors difficult to detect during manufacturing test.
    - Example: a Trojan might cause an error detection module to accept inputs that should be rejected

Hardware Trojan Horses: Payload

- **Downgrade Performance:** A Trojan downgrades performance by intentionally changing device parameters such as power and delay.
  - Example: a Trojan might insert more buffers in the chip’s interconnections and, hence, consume more power, which in turn could drain the battery quickly.

Hardware Trojan Horses: Payload

- **Leak Information:** A Trojan can leak information through both covert and overt channels. Sensitive data can be leaked via radio frequency, optical or thermal power, timing side-channels, and interfaces, such as RS-232 and JTAG
  - Example, a Trojan might leak a cryptographic algorithm's secret key through unused RS-232 ports.

Hardware Trojan Horses: Payload

- **Denial-of-Service:** A denial-of-service (DoS) Trojan can cause the target module to exhaust scarce resources, such as bandwidth, computation, and battery power
  - Example a Trojan alter the device’s configuration causing a processor to ignore the interrupt from a specific peripheral. DoS can be either temporary or permanent.

Hardware Trojan Horses
-
Avoidance, Detection and Tolerance

Why is detection of hardware Trojans very difficult?

- **Verification (Traditional)**
  - Bugs (Unintentional)
  - Bounded by Spec
- **Trust Verification**
  - Malicious change (**Intentional**)
  - Unwanted functionality (**Unbounded**)

Trojan Attacks → BIGGER verification challenge!

Why is detection of hardware Trojans very difficult?

HW Trojans are meant to stay silent most of the time…

- Small size
- Reduce power and EM footprint

HW Detection: Side-channel Analysis

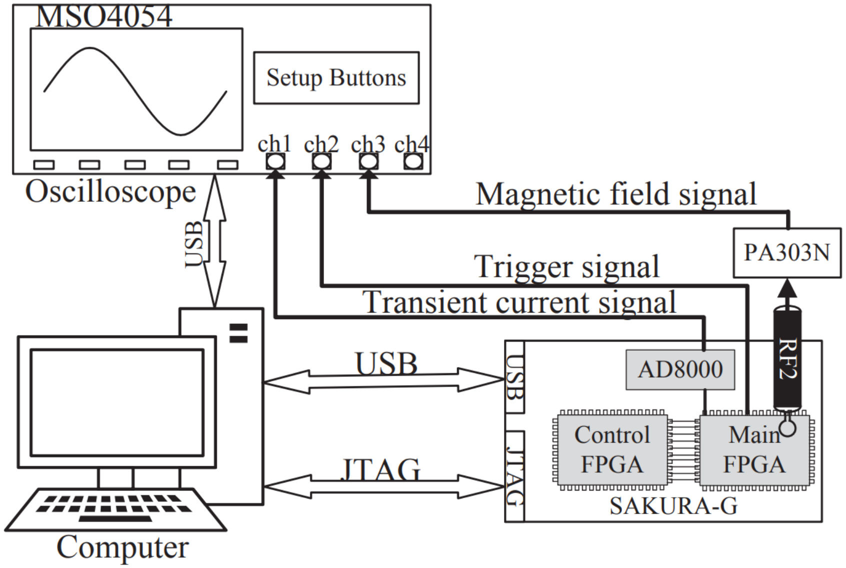
Applied at:

- after manufacturing

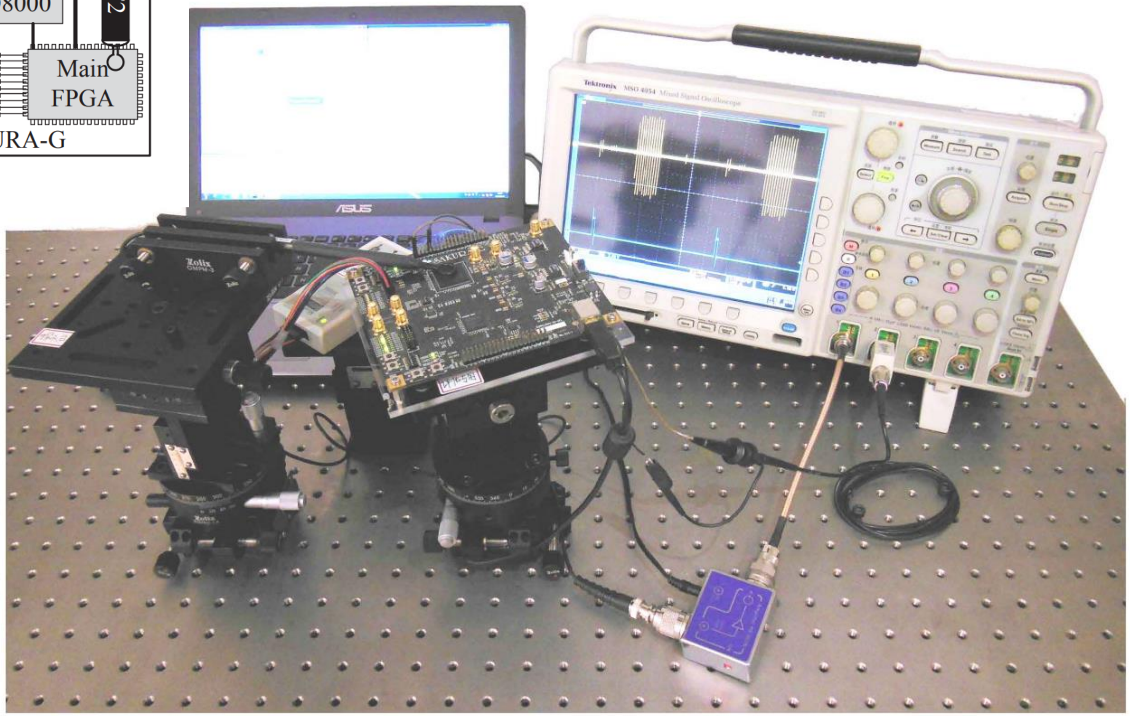
Considered attacker:

- foundry

HW Detection: Side-channel Analysis



Study EM emissions, power consumption, timing…



HW Detection: Side-channel Analysis

Limitations:

- Need for a *golden* prototype (either physical or modelled)
  - to identify the refence values
- Need for activating the Trojan during the analysis
- Made difficult by process variation

Y. Liu, Y. Zhao, J. He, A. Liu and R. Xin, "SCCA: Side-channel correlation analysis for detecting hardware Trojan," 2017 11th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID), Xiamen, China, 2017, pp. 196-200.

J. He, Y. Zhao, X. Guo and Y. Jin, "Hardware Trojan Detection Through Chip-Free Electromagnetic Side-Channel Statistical Analysis," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 10, pp. 2939-2948, Oct. 2017.

J. Cruz, F. Farahmandi, A. Ahmed and P. Mishra, "Hardware Trojan Detection Using ATPG and Model Checking," 2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID), Pune, India, 2018, pp. 91-96

HW Detection: Logic testing / Circuit analysis

Applied at:

- logic design *
- physical design *
- after manufacturing *

Considered attacker:

- IP provider *
- employees **
- CAD tool **
- foundry *

HW Detection: Logic testing / Circuit analysis

- Generate test patterns to activate the Trojan
- Analyze the switching activity of wires and gates in the circuit
  - Identify unused or rarely used resources

HW Detection: Logic testing / Circuit analysis

Limitations:

- Need for a *golden* prototype (either physical or modelled)
  - to identify the expected output values
- Need for activating the Trojan during the analysis
- Controllability and observability

H. Salmani and M. Tehranipoor, "Layout-Aware Switching Activity Localization to Enhance Hardware Trojan Detection," in *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 76-87, Feb. 2012

H. Salmani and M. Tehranipoor, "Analyzing circuit vulnerability to hardware Trojan insertion at the behavioral level," *2013 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, New York, NY, USA, 2013, pp. 190-195

F. S. Hossain, M. Shintani, M. Inoue and A. Orailoglu, "Variation-Aware Hardware Trojan Detection through Power Side-channel," 2018 IEEE International Test Conference (ITC), Phoenix, AZ, USA, 2018, pp. 1-10.

HW Detection: Formal modeling and analysis

Applied at:

- logic design

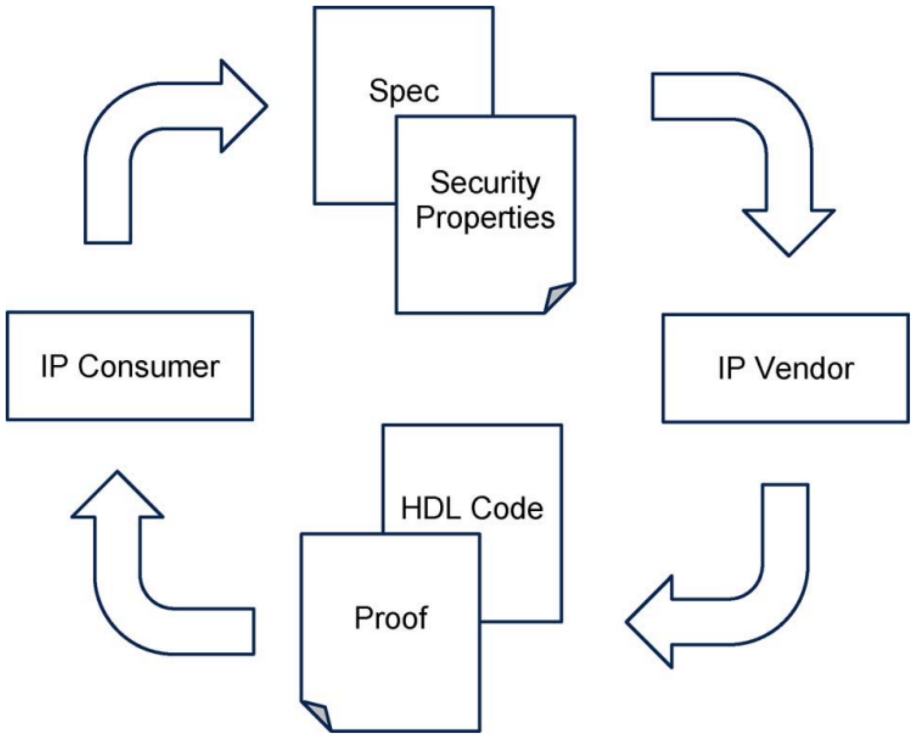
Considered attacker:

- IP provider
- employees
- CAD tool

HW Detection: Formal modeling and analysis

Formalize properties, circuits, systems, interactions…

Use theorem-proving and model-checking for verification



HW Detection: Formal modeling and analysis

Limitations:

- Need to *capture* possible ‘‘under-attack’’ behaviors
- Difficult to model all possible working conditions
- Scalability

M. Rathmair, F. Schupfer and C. Krieg, "Applied formal methods for hardware Trojan detection," 2014 IEEE International Symposium on Circuits and Systems (ISCAS), Melbourne, VIC, Australia, 2014, pp. 169-172

E. Love, Y. Jin and Y. Makris, "Proof-Carrying Hardware Intellectual Property: A Pathway to Trusted Module Acquisition," in IEEE Transactions on Information Forensics and Security, vol. 7, no. 1, pp. 25-40, Feb. 2012

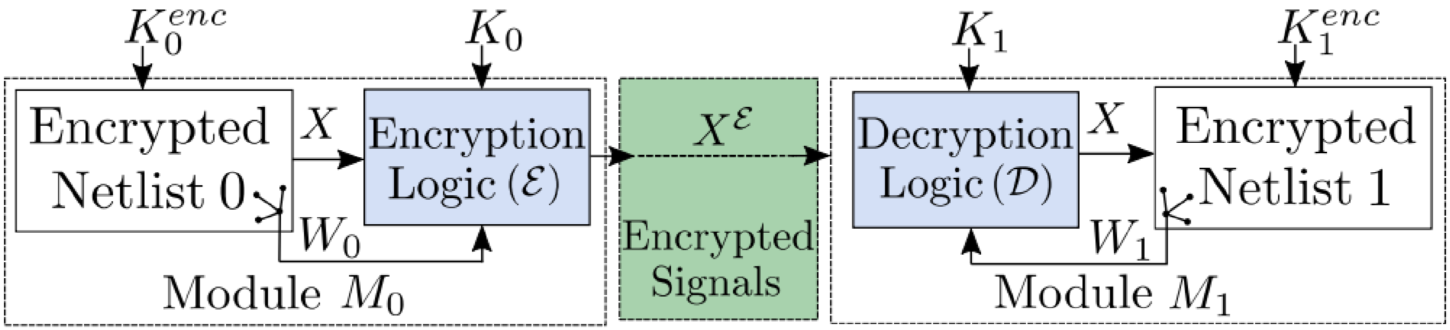
Hardware Trojan Horses
-
Design for Trust

Hardware Trojan Horses: Design for Trust

It is *impossible* to guarantee the absence of HW Trojans in the system

The *new trend* is building trusted system with untrusted components or ensure trusted execution on untrusted system

Hardware Trojan Horses: Design for Trust



Applied at:

- System integration-level

Considered attacker:

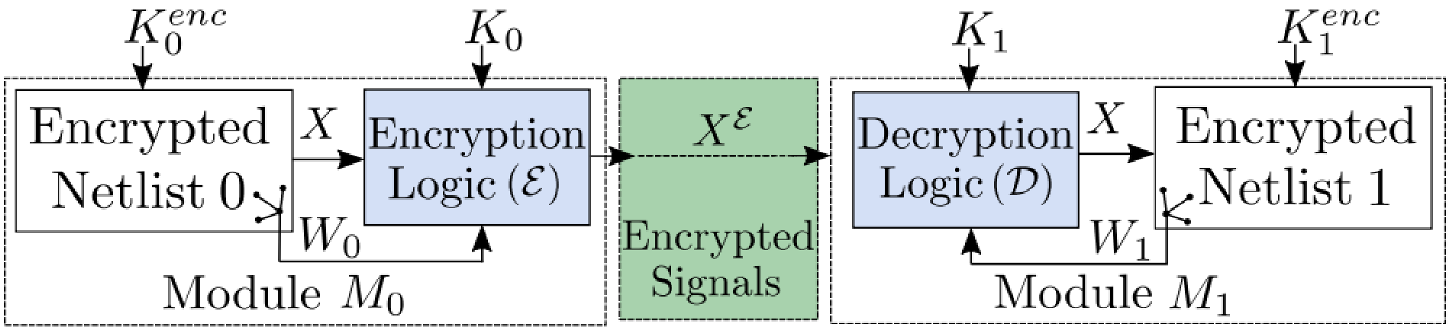
- IP provider

Type of attack:

- Any trojan that is activated by external signals

Šišejković, Dominik, et al. "Control-lock: Securing processor cores against software-controlled hardware trojans." Proceedings of the 2019 on Great Lakes Symposium on VLSI. 2019.

Hardware Trojan Horses: Design for Trust



Solution:

- Encrypt communications between IP cores

Limitations:

- Does not consider always-on and internally activated Trojans

Hardware Trojan Horses: Design for Trust

Applied at:

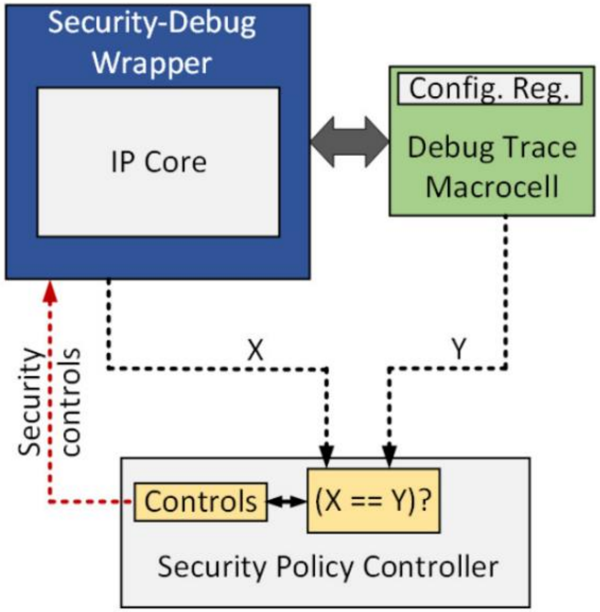
- System integration-level

Considered attacker:

- IP provider

Type of attack:

- Any trojan that tries to interact with other modules

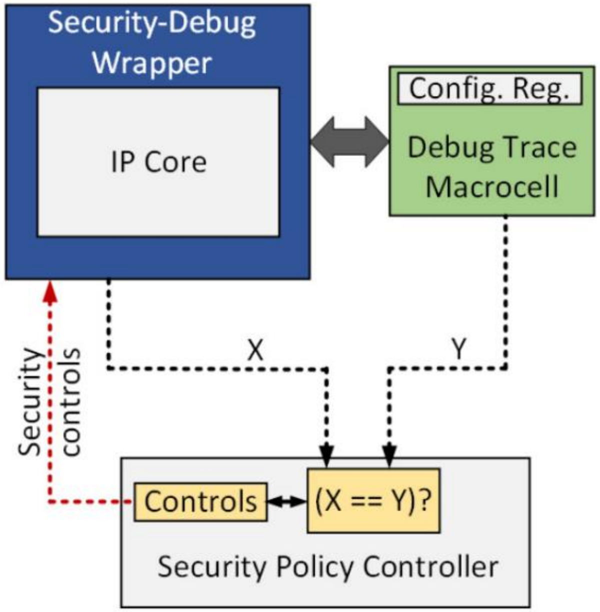


A. Basak, S. Bhunia, T. Tkacik and S. Ray, "Security Assurance for System-on-Chip Designs With Untrusted IPs," in *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 7, pp. 1515-1528, July 2017

Hardware Trojan Horses: Design for Trust

Solution:

- Implement ad hoc HW firewall, one for each untrusted IP core



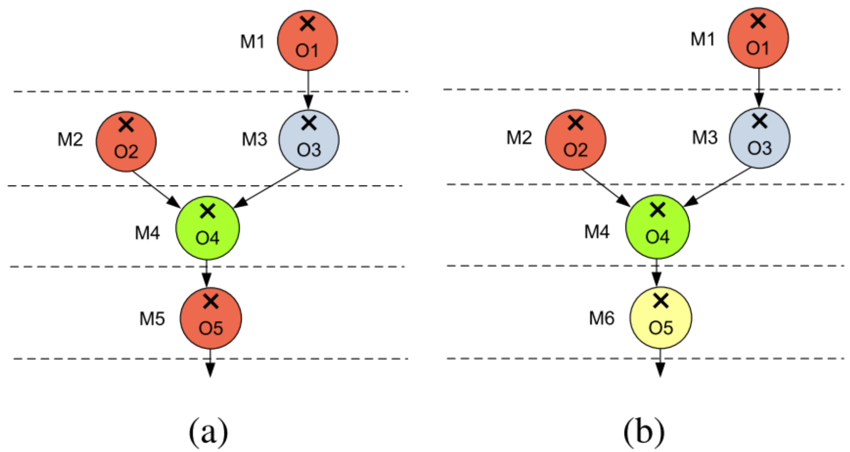
Limitations:

- Does not protect against HW Trojans that do not communicate with separate modules

Hardware Trojan Horses: Design for Trust

Applied at:

- System integration-level / high-level syntesis



Considered attacker:

- IP provider

Type of attack:

- Any trojan that tries to interact with other modules

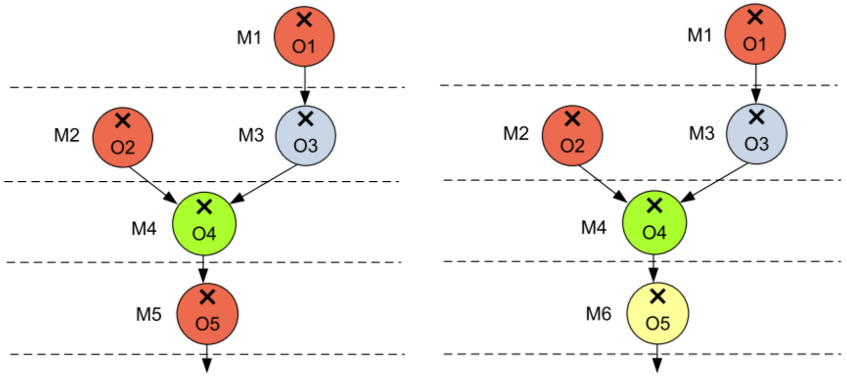
C. Liu, J. Rajendran, C. Yang and R. Karri, "Shielding heterogeneous MPSoCs from untrustworthy 3PIPs through security-driven task scheduling," *2013 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, New York, NY, USA, 2013, pp. 101-106

J. J. Rajendran, O. Sinanoglu and R. Karri, "Building Trustworthy Systems Using Untrusted Components: A High-Level Synthesis Approach," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 9, pp. 2946-2959, Sept. 2016

Hardware Trojan Horses: Design for Trust

Solution:

- Modules belonging to the same vendor are not placed on the same *execution path*



(a) (b)

Limitations:

- Does not protect against HW Trojans that do not communicate with separate modules

Hardware Trojan Horses
-
Design for Trust @POLIMI

Who Is involved

- Elia Lazzeri
- Nikita Litovchenko
- Alessandro Nazzari
- Mattia Iamundo
- Tomas Antonio Lopez

- Giorgio Di Natale

- Giuseppe Bianchi
- Marco Ottavi
- Alessandro Palumbo*
- Salvatore Pontarelli*

- José Hernández
- Pedro Reviriego*

- Gianluca Furano

- Marco Ottavi

- Kerem Arıkan*
- Alperem Bolat*
- Oguz Ergin

- Alessandra Menicucci
- Stefano Di Mascio

* Currently Alessandro is with Politecnico di Milano, Salvatore with Università di Roma "La Sapienza", Pedro with Uiversidad Politecnica de Madrid, Kerem with Binghamton University and Alperem with Queen University Belfast

Hardware Trojan Horses: Design for Trust

Applied at:

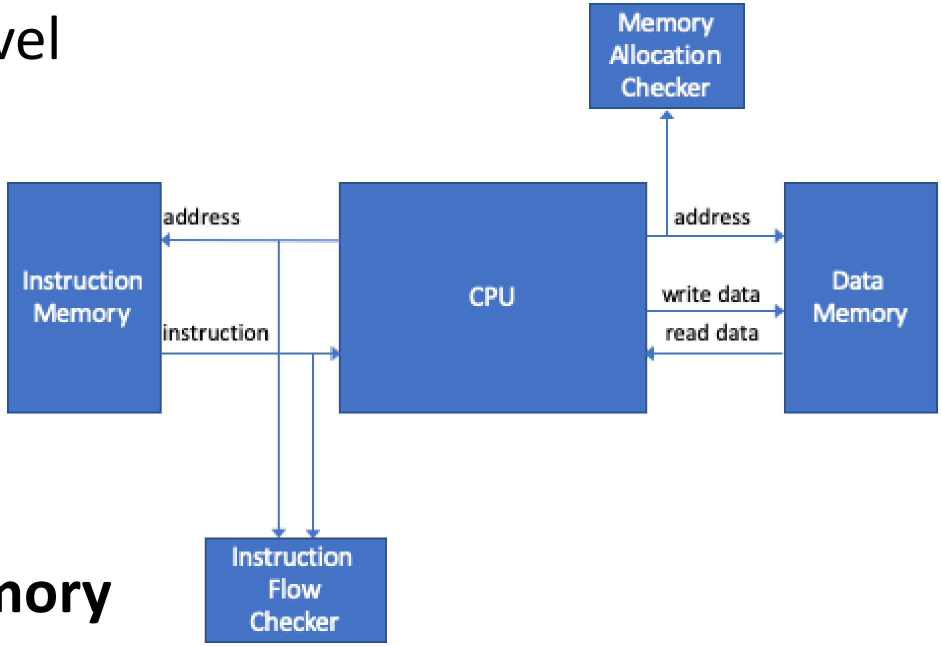
- System integration-level

Considered attacker:

- IP provider

Type of attack:

- Any trojan **in the memory** that tries to force the execution of unexpected instructions or to access unauthorized locations

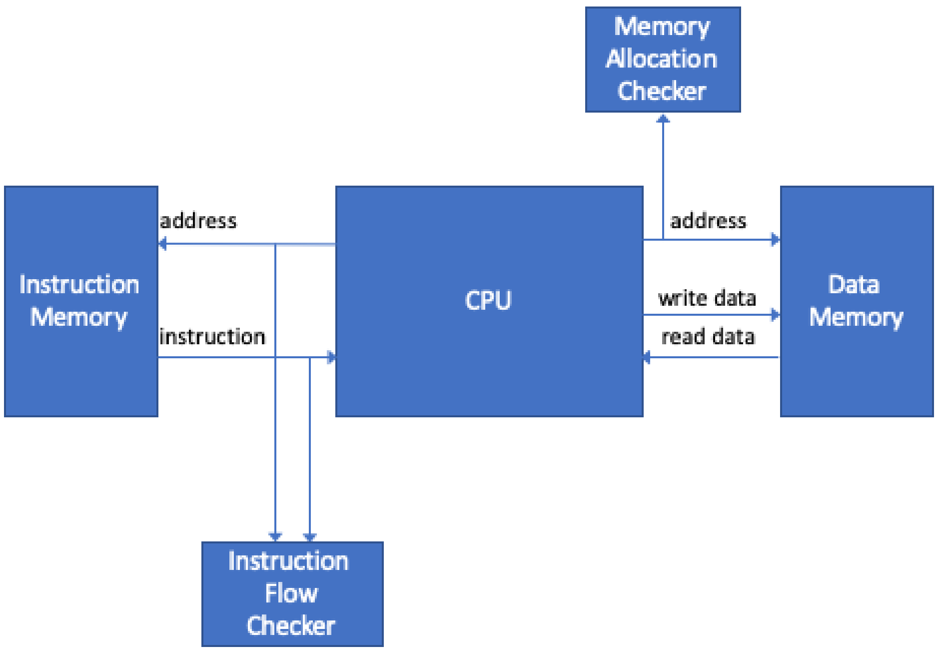


A. Bolat, L. Cassano, P. Reviriego, O. Ergin and M. Ottavi, "A Microprocessor Protection Architecture against Hardware Trojans in Memories," *2020 15th Design & Technology of Integrated Systems in Nanoscale Era (DTIS)*, Marrakech, Morocco, 2020, pp. 1-6

Hardware Trojan Horses: Design for Trust

Solution:

- Bloom filter-based security checkers added on the instruction and data buses



Limitations:

- Does not protect against HW Trojans that perform information stealing and Denial-of-Sevice

Hardware Trojan Horses: Design for Trust

A Bloom filter is a data structure designed to tell you, rapidly and memory-efficiently, whether an element is present in a set.

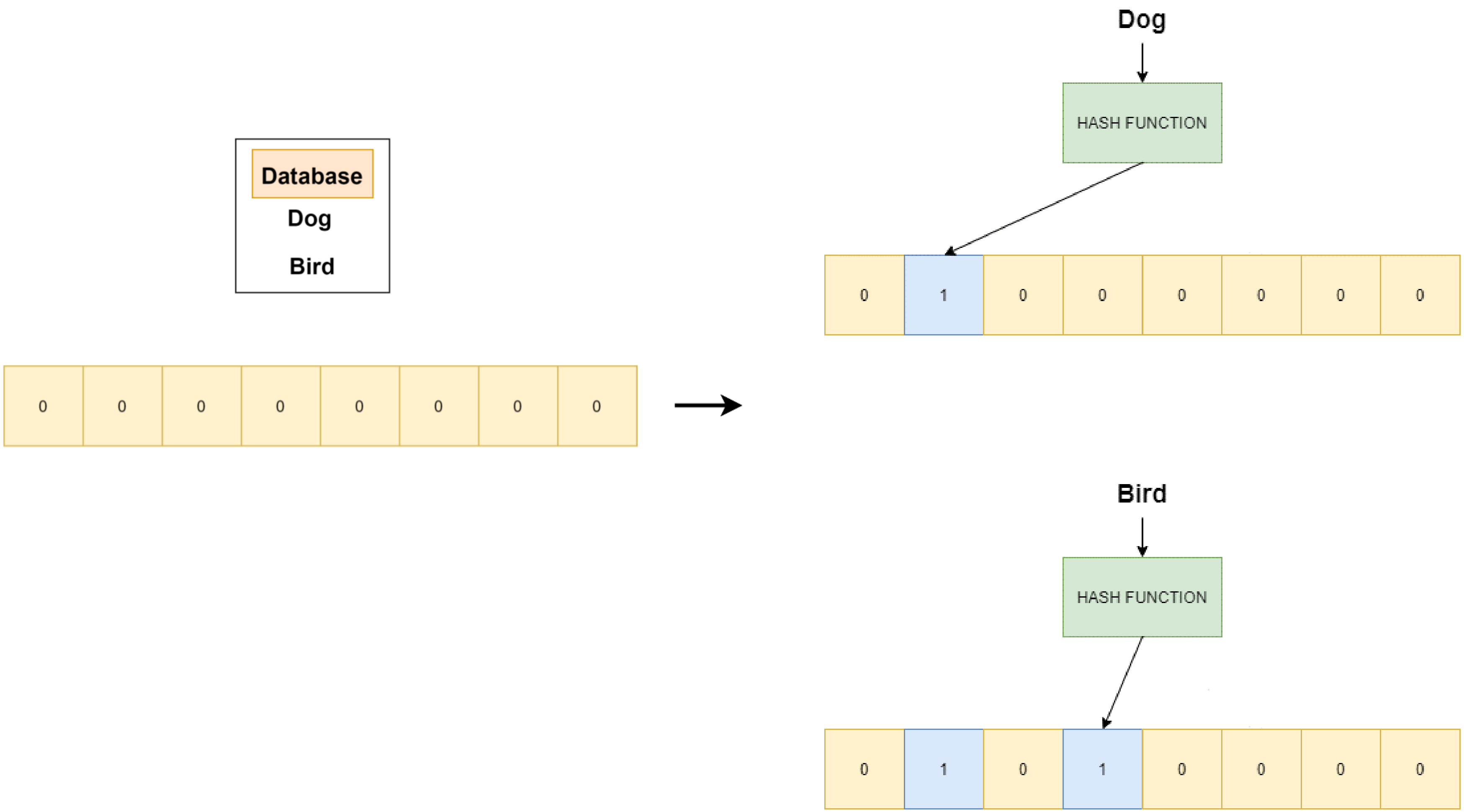
The base components of a Bloom filter are Bit Vectors and Hash functions

Bloom Filter may be:

- Configured
- Queried

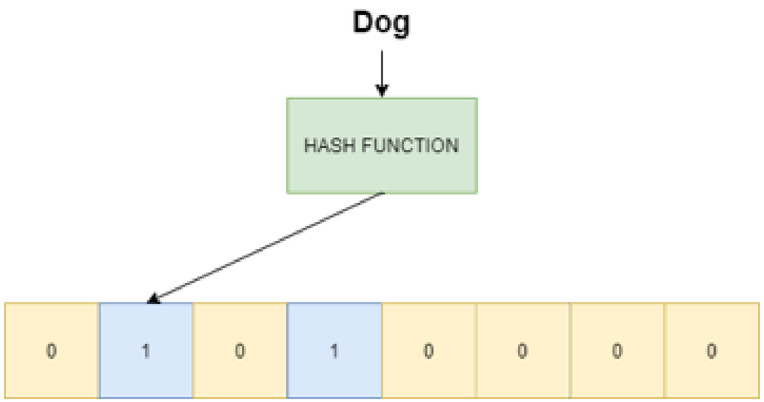
Hardware Trojan Horses: Design for Trust

Bloom filter configuration

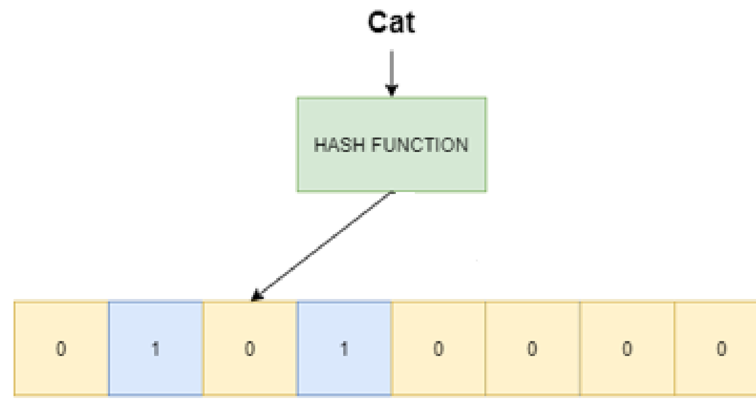


Hardware Trojan Horses: Design for Trust

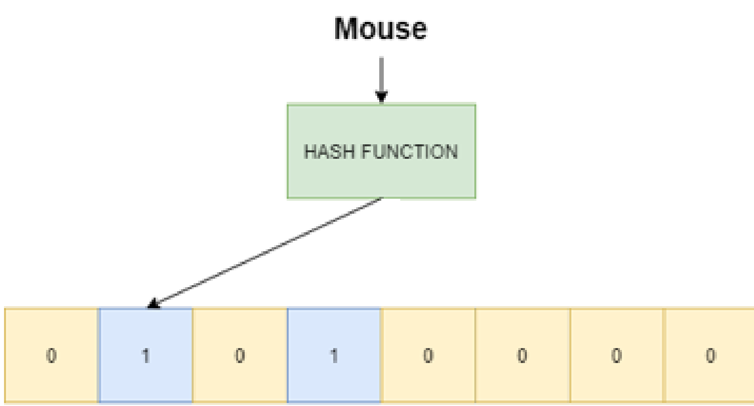
Bloom filter querying



True positive



True negative



False positive (hash collision)

Hardware Trojan Horses: Design for Trust

Bloom filter: how to control the false positive rate?

Solution 1: increase the size of the bit array

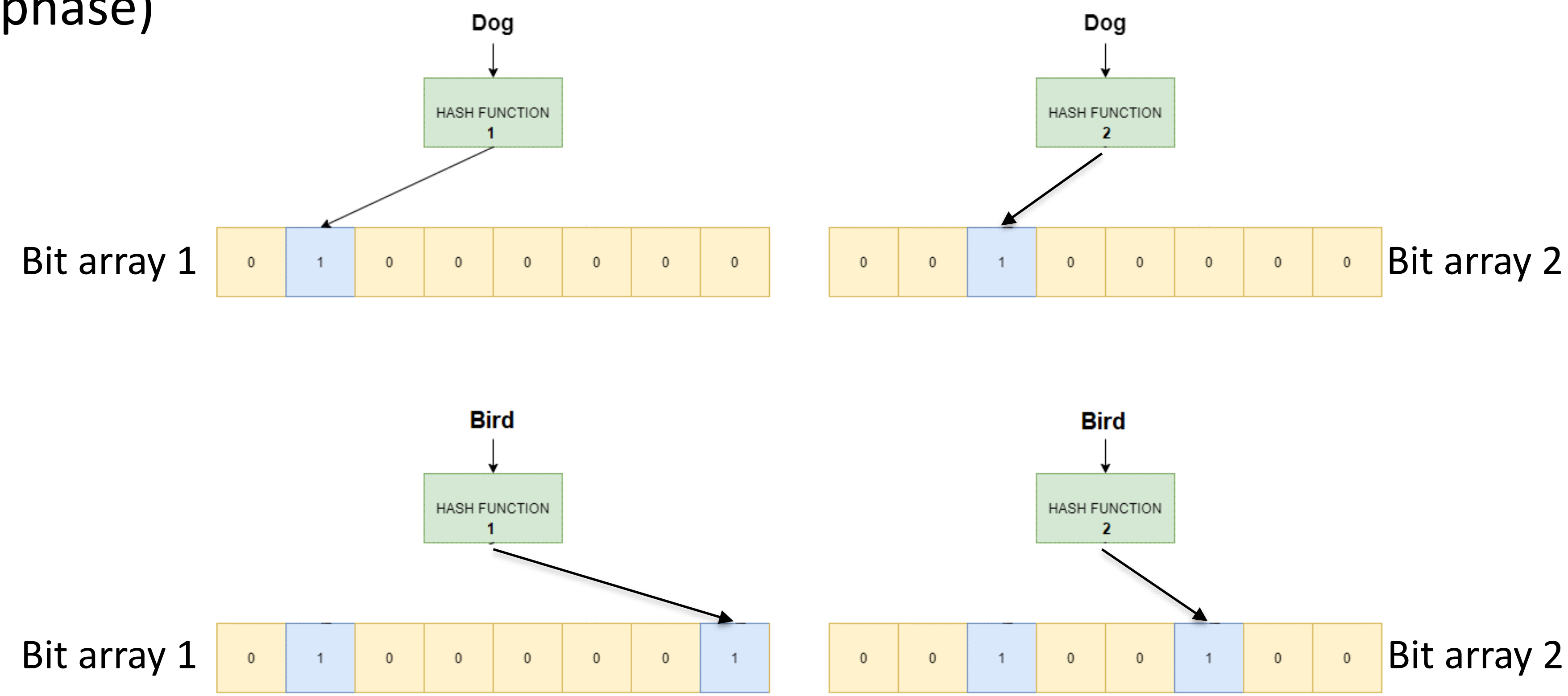
0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Hardware Trojan Horses: Design for Trust

Bloom filter: how to control the false positive rate?

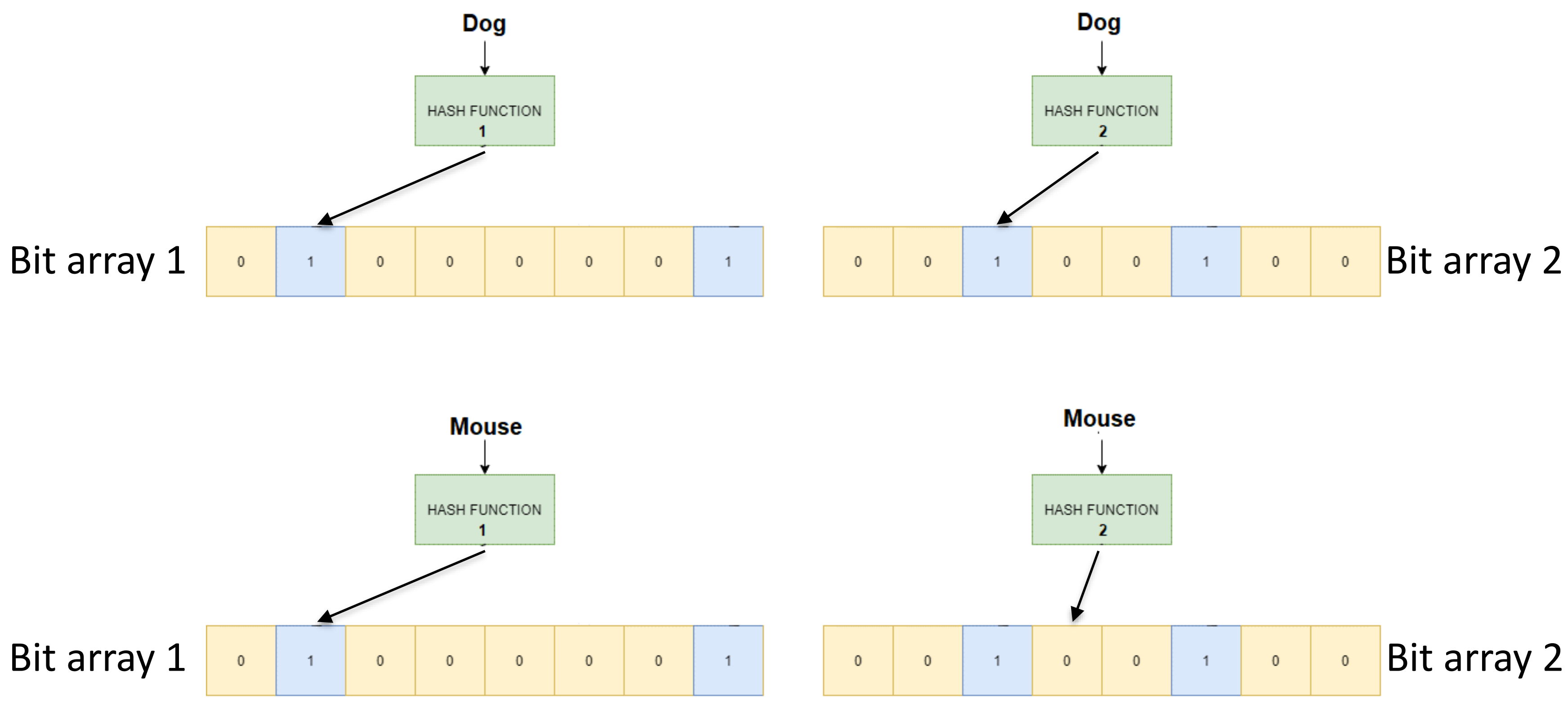
Solution 2: increase the number of hash functions (Configuration phase)



Hardware Trojan Horses: Design for Trust

Bloom filter: how to control the false positive rate?

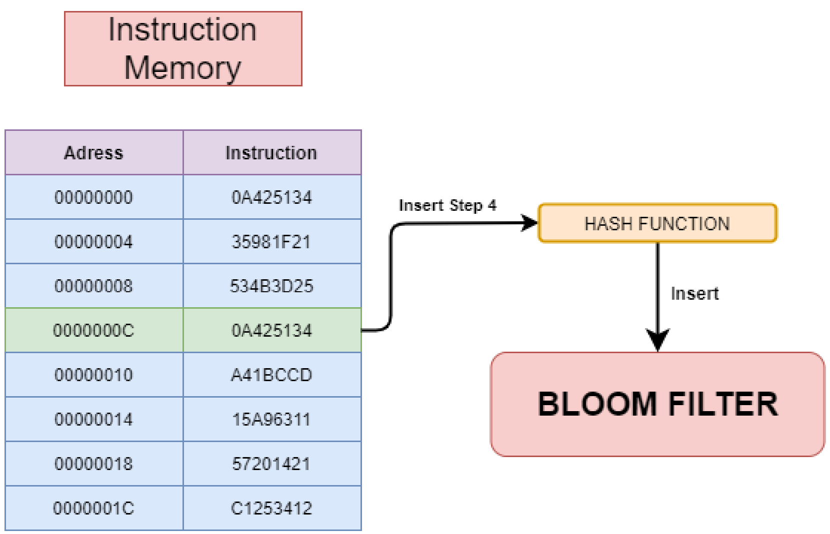
Solution 2: increase the number of hash functions (Query phase)



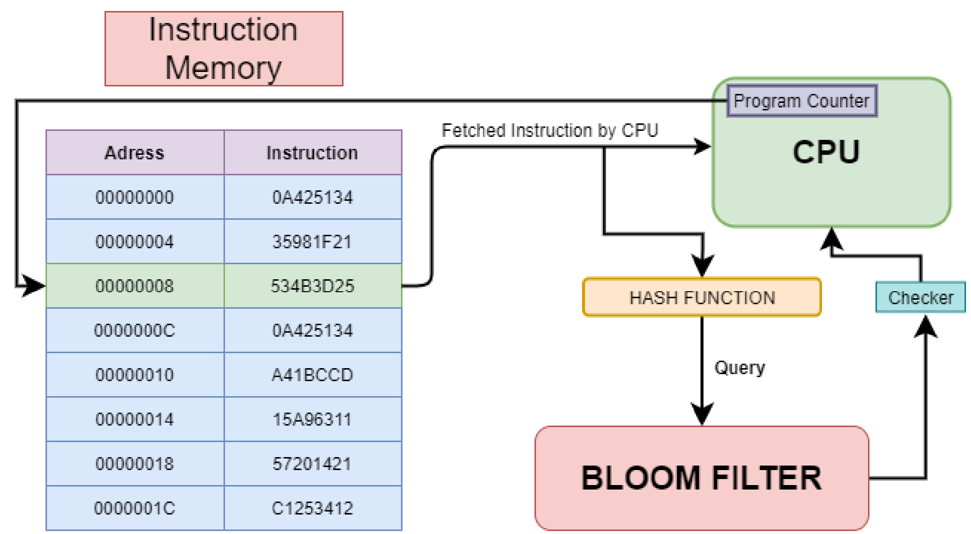
Hardware Trojan Horses: Design for Trust

The proposed solution

Configuration (program installation time)



Query (program execution time)



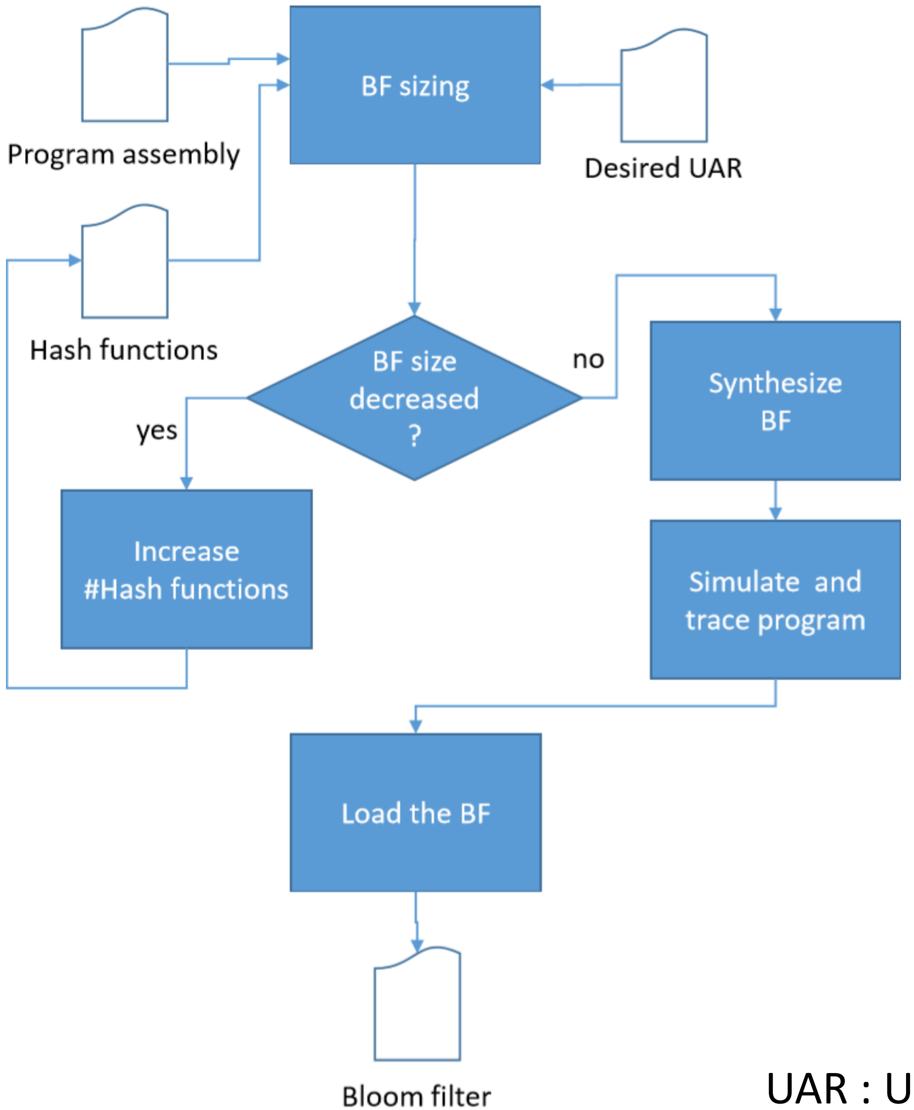
Hardware Trojan Horses: Design for Trust

If the number of hash functions increases the size of the associated bit arrays can be decreased and viceversa

The best configuration in terms of number of hash functions and size of the bit arrays has to be found

- Achieve a desired *undetected alarms rate*
- Minimize area overhead

Hardware Trojan Horses: Design for Trust



UAR : Undetected Alarms Rate

Hardware Trojan Horses: Design for Trust

SETUP

- We set 1% as a target Undetected Alarms Rate
- Modelsim for instructions lists and emory traces of benchmark programs.
- Xilinx Artix XC7A35T as a target prototyping device
- RI5CY Pulpino as a target CPU
  - 14616 LUTs
  - 8959 FFs
  - 16 BRAMs
  - working 50MHz

Benchmarks	Instruction Number
Binary Sort	1719
Matrix Multiplication	1733
Bubble Sort	1775
Quick Sort	1927
Sudoku Solver	3227
Motion Detection	4452

PULPino

Hardware Trojan Horses: Design for Trust

We tried several number of hash functions and the best configuration was with 5

SIZE (IN KBIT) AND ACHIEVED UAR OF THE HARDWARE IMPLEMENTATION OF THE BFS FOR THE IFC AND THE MAC

Bench.	IFC		MAC	
	Mem. size	UAR (%)	Mem. size	UAR (%)
BinS	32	0.523	1	0.085
MM	32	0.520	1	0.122
BubS	32	0.572	1	0.525
QS	32	0.607	1	0.073
SD	64	0.249	4	0.134
MD	64	0.912	8	0.232

Hardware Trojan Horses: Design for Trust

RESOURCE OCCUPATION AND WORKING FREQUENCY OF THE IMPLEMENTED IFCS

Bench.	Instruction Flow Checker		
	#LUTs	#FFs	Freq. (MHz)
BinS	880 (6.02%)	84 (0.93%)	112.19
MM	880 (6.02%)	84 (0.93%)	112.19
BubS	880 (6.02%)	84 (0.93%)	112.19
QS	880 (6.02%)	84 (0.93%)	112.19
SS	1539 (10.52%)	89 (0.99%)	106.37
MD	1539 (10.52%)	89 (0.99%)	106.37

RESOURCE OCCUPATION AND WORKING FREQUENCY OF THE IMPLEMENTED IFCS

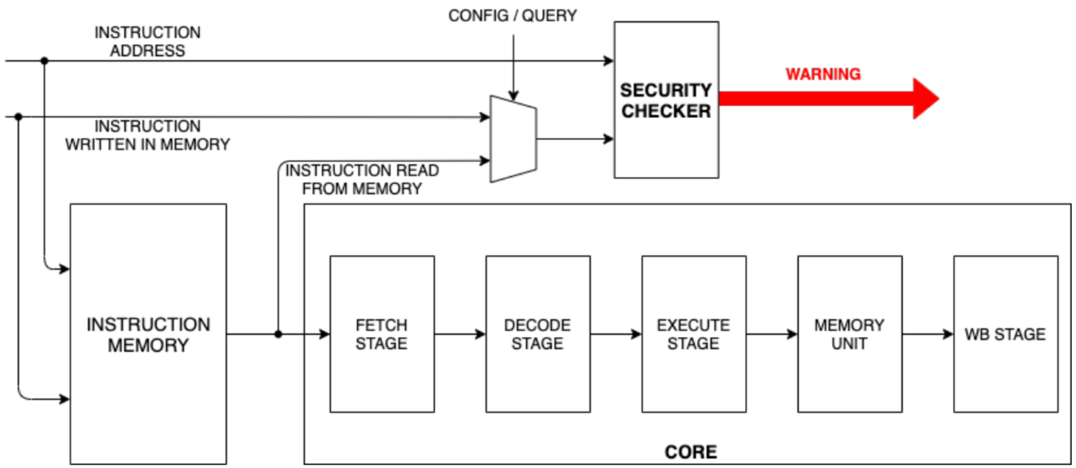
Bench.	Instruction Flow Checker		
	#LUTs	#FFs	Freq. (MHz)
BinS	880 (6.02%)	84 (0.93%)	112.19
MM	880 (6.02%)	84 (0.93%)	112.19
BubS	880 (6.02%)	84 (0.93%)	112.19
QS	880 (6.02%)	84 (0.93%)	112.19
SS	1539 (10.52%)	89 (0.99%)	106.37
MD	1539 (10.52%)	89 (0.99%)	106.37

- Area overhead from 7% up to 12%
- No working frequency reduction

Hardware Trojan Horses: Design for Trust

Applied at:

- System integration-level



Considered attacker:

- IP provider

Type of attack:

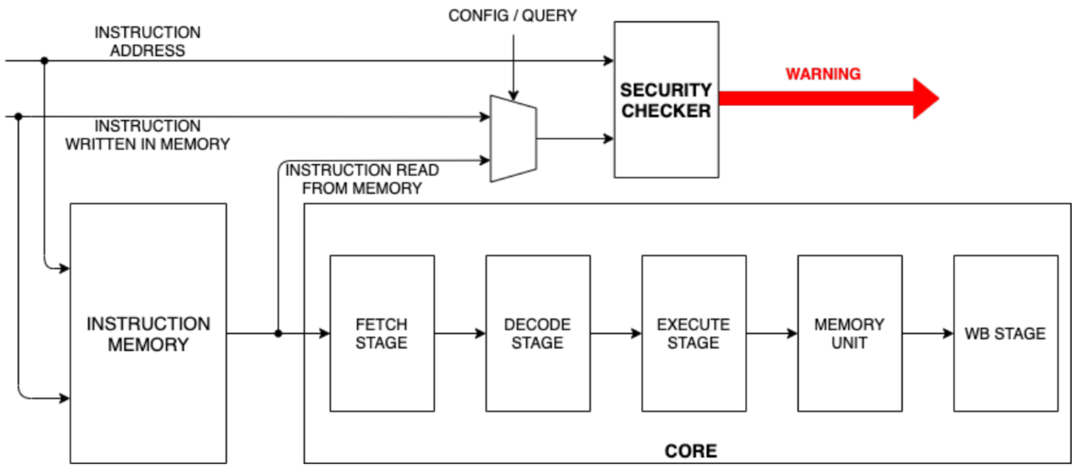
- Any trojan **in the CPU** that tries to force the execution of unexpected instructions or to access unauthorized locations

A. Palumbo, L. Cassano, P. Reviriego, G. Bianchi and M. Ottavi, "A Lightweight Security Checking Module to Protect Microprocessors against Hardware Trojan Horses," 2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Athens, Greece, 2021, pp. 1-6

Hardware Trojan Horses: Design for Trust

Solution:

- Security checkers added on the system bus



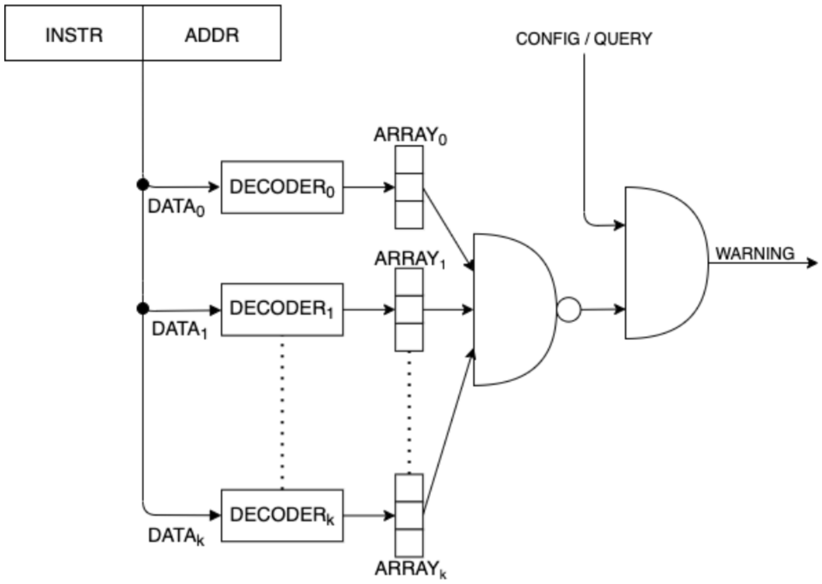
Limitations:

- Does not protect against HW Trojans that perform information stealing and Denial-of-Sevice

Hardware Trojan Horses: Design for Trust

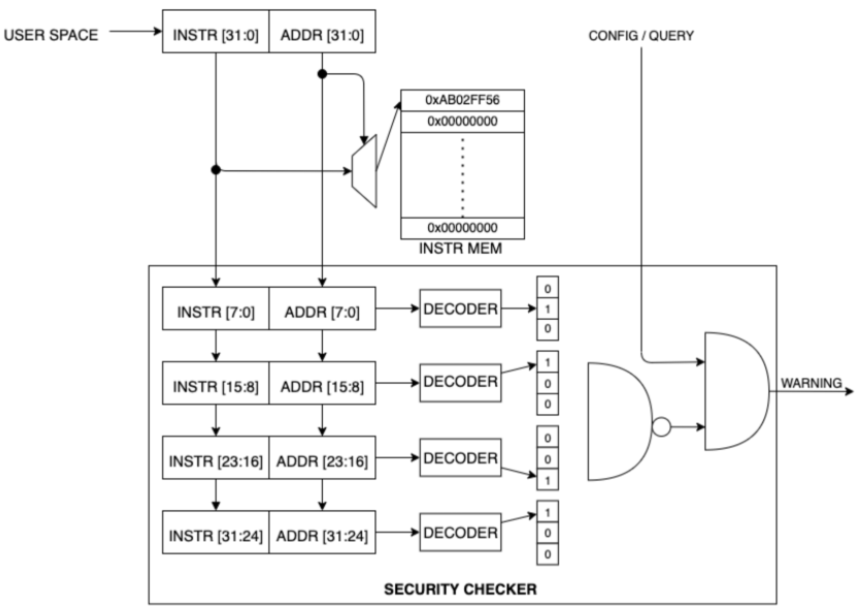
Hash functions are substituted by *simple* decoders

Instruction+Operand (part of this) address are used as addresses to access the bit arrays

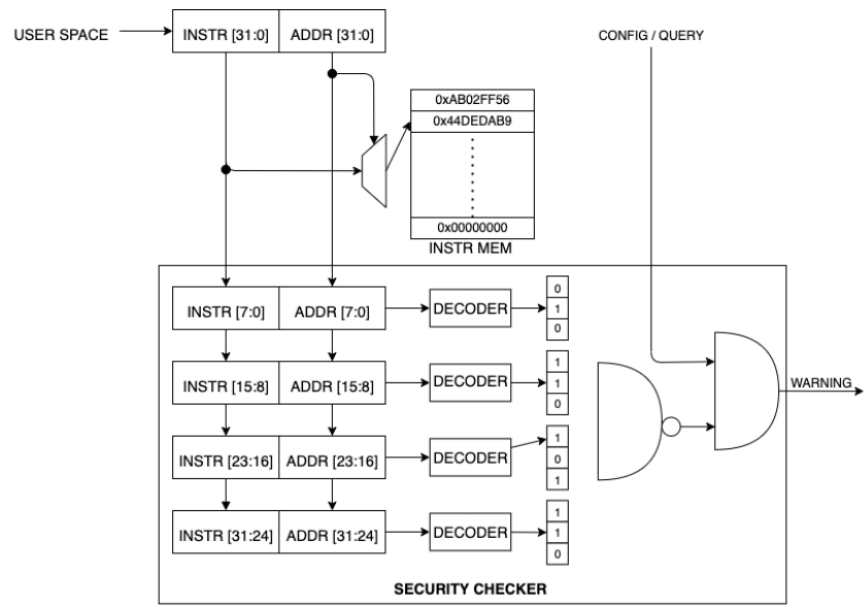


Hardware Trojan Horses: Design for Trust

Configuration



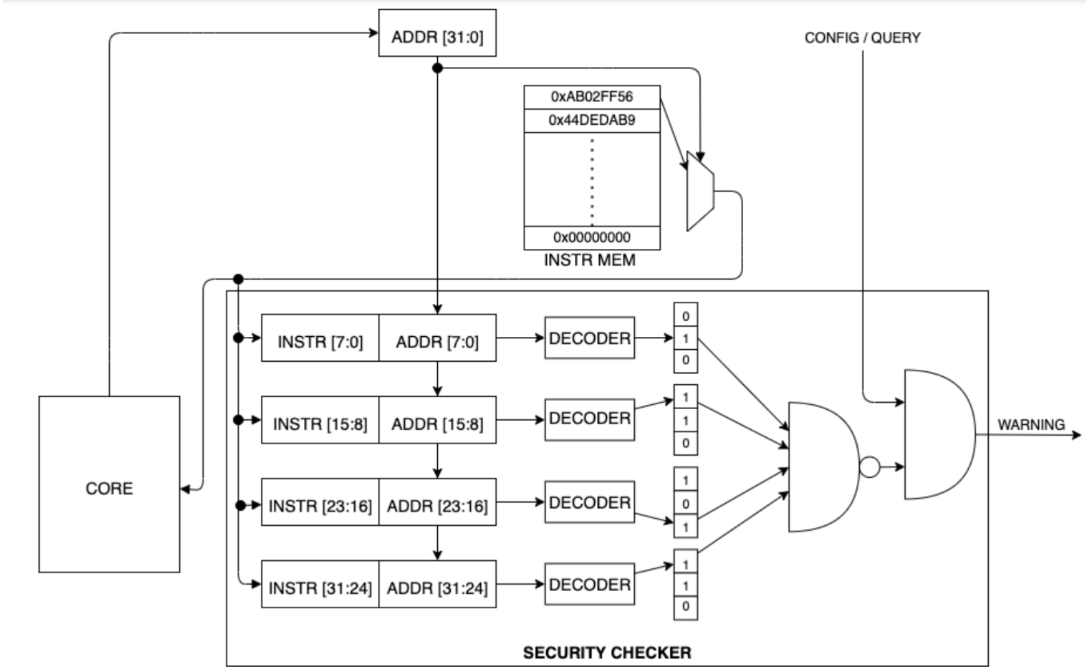
(a) Writing the first program instruction



(b) Writing the second program instruction

Hardware Trojan Horses: Design for Trust

Query



Hardware Trojan Horses: Design for Trust

We considered the same setup as the previous solution

- We set 1% as a target Undetected Alarms Rate
- Modelsim for instructions lists and emory traces of benchmark programs.
- Xilinx Artix XC7A35T as a target prototyping device
- RI5CY Pulpino as a target CPU
  - 14616 LUTs
  - 8959 FFs
  - 16 BRAMs
  - working 50MHz

Benchmarks	Instruction Number
Binary Sort	1719
Matrix Multiplication	1733
Bubble Sort	1775
Quick Sort	1927
Sudoku Solver	3227
Motion Detection	4452

PULPino

Hardware Trojan Horses: Design for Trust

Table II: FP and FN rates when the HTH modifies the accessed instruction memory location

Bench.	Our proposal		Proposal in [26]	
	FP	FN	FP	FN
BinS	0%	0%	0%	0.523%
MM	0%	0%	0%	0.520%
BubS	0%	0%	0%	0.572%
QS	0%	0%	0%	0.607%
SD	0%	0%	0%	0.249%
MD	0%	0%	0%	0.912%
AVG	0%	0%	0%	0.663%

Table IV: FP and FN rates when the HTH modifies the fetched instruction

Bench.	Accuracy	
	FP	FN
BinS	0%	2.25%
MM	0%	0.40%
BubS	0%	3.01%
QS	0%	3.91%
SD	0%	0.72%
MD	0%	2.83%
AVG	0%	2.18%

Hardware Trojan Horses: Design for Trust

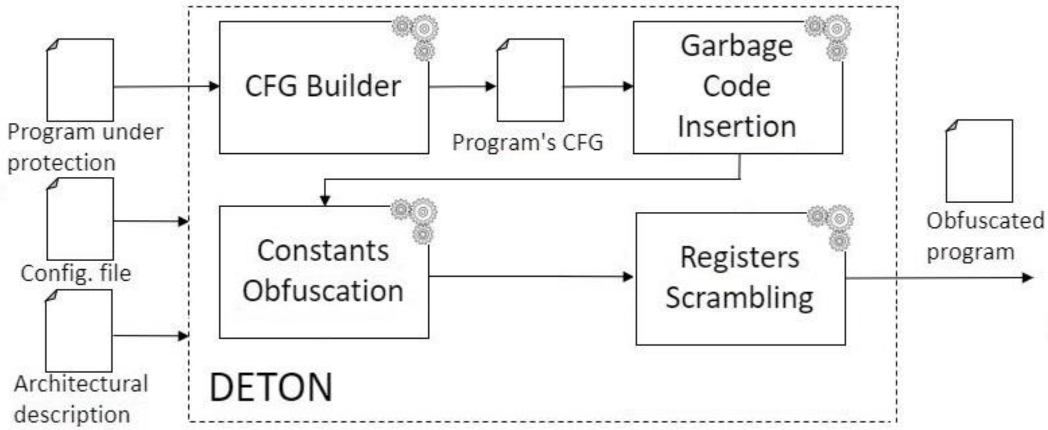
Table III: Resource occupation and working frequency of our proposal and of the one in [26]

Bench.	Our proposal				Proposal in [26]			
	#LUTs	#FFs	BRAM size	Freq. (MHz)	#LUTs	#FFs	BRAM size	Freq. (MHz)
BinS	75 (0.49%)	31 (0.31%)	208 Kbit	275 MHz	880 (5.83%)	84 (0.85%)	32 KBit	112 MHz
MM	75 (0.49%)	31 (0.31%)	208 Kbit	275 MHz	880 (5.83%)	84 (0.85%)	32 KBit	112 MHz
BubS	75 (0.49%)	31 (0.31%)	208 Kbit	275 MHz	880 (5.83%)	84 (0.85%)	32 KBit	112 MHz
QS	75 (0.49%)	31 (0.31%)	208 Kbit	275 MHz	880 (5.83%)	84 (0.85%)	32 KBit	112 MHz
SS	75 (0.49%)	31 (0.31%)	208 Kbit	275 MHz	1539 (10.19%)	89 (0.90%)	64 KBit	106 MHz
MD	75 (0.49%)	31 (0.31%)	208 Kbit	275 MHz	1539 (10.19%)	89 (0.90%)	64 KBit	106 MHz

Hardware Trojan Horses: Design for Trust

Applied at:

- Compile-time (the system is already deployed)



Considered attacker:

- IP provider

Type of attack:

- Information-stealing HTHs that periodically monitor a number of registers of the CPU and transmits data

- DETON: DEfeating hardware Trojan horses in microprocessors through software ObfuscatioN, Luca Cassano *et al.*, In Journal of Systems Architecture, vol. 129, 2022
- On the optimization of Software Obfuscation against Hardware Trojans in Microprocessors, Luca Cassano *et al.*, In DDECS 2022

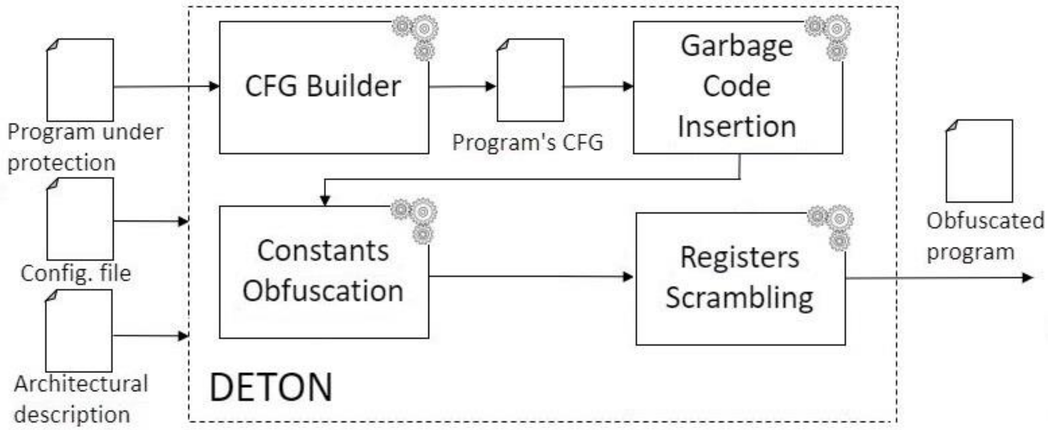
Hardware Trojan Horses: Design for Trust

Solution:

- Compile time SW obfuscation

Limitations:

- Does not protect against HW Trojans that perform change functionality and Denial-of-Sevice
- Does not protect against *immediate* information stealing



Hardware Trojan Horses: Design for Trust

Garbage code insertion

- Random insertion point
- Random instructions
    - No division and no jump instructions
- Random sequence length

```
main:
        addi    sp, sp,-32
        sd      s0, 24(sp)
        addi    s0, sp, 32
        li      a5, 10
        sw      a5, -20(s0)
        li      a5, 4
        sw      a5, -24(s0)
        lw      a5, -20(s0)
        addiw   a5, a5, 1
        sw      a5, -20(s0)
        lw      a4, -24(s0)
        lw      a5, -20(s0)
        addw    a5, a4, a5
        sw      a5, -24(s0)
        li      a5, 0
        mv      a0, a5
        ld      s0, 24(sp)
        addi    sp, sp, 32
        jr      ra
```

```
slli    a6, t5, 30
sltu    a4, s3, s11
mv      t4, a7
mulh    t4, s5, a1
sltiu   t0, ra, -1657
sra     s11, t6, s5
```

Hardware Trojan Horses: Design for Trust

Constant value obfuscation

- Substitute a constant value with an instruction sequence that produces the constant value

```
main:
        addi   sp, sp,-32
        sd     s0, 24(sp)
        addi   s0, sp, 32
        li     a5, 10
        sw     a5, -20(s0)
        li     a5, 4
        sw     a5, -24(s0)
        lw     a5, -20(s0)
        addiw  a5, a5, 1
        sw     a5, -20(s0)
        lw     a4, -24(s0)
        lw     a5, -20(s0)
        addw   a5, a4, a5
        sw     a5, -24(s0)
        li     a5, 0
        mv     a0, a5
        ld     s0, 24(sp)
        addi   sp, sp, 32
        jr     ra
```

```
li     t5, 463
slli   s6, t5, 3
srli   s9, s6, 3
ori    t2, s9, -976
slli   t3, t2, 0
andi   a7, t3, 1019
andi   t6, a7, -450
ori    t0, t6, 1668
andi   a2, t0, -1727
ori    t1, a2, 1
slli   t4, t1, 11
srli   s11, t4, 6
add    s0, sp, s11
```

Hardware Trojan Horses: Design for Trust

Register scrambling

- Periodically move the content of a register from a register to an unused register

```
main:
        addi    sp, sp,-32
        sd      s0, 24(sp)
        addi    s0, sp, 32
        li      a5, 10
        sw      a5, -20(s0)
        li      a5, 4
        sw      a5, -24(s0)
        lw      a5, -20(s0)
        addiw   a5, a5, 1
        sw      a5, -20(s0)
        lw      a4, -24(s0)
        lw      a5, -20(s0)
        addw    a5, a4, a5
        sw      a5, -24(s0)
        li      a5, 0
        mv      a0, a5
        ld      s0, 24(sp)
        addi    sp, sp, 32
        jr      ra
```

```
addi    sp, sp,-32
mv      t1, sp
sd      s0, 24(t1)
addi    s0, t1, 32
```

Hardware Trojan Horses: Design for Trust

Some results

We implemented DETON as a set of C programs and Python scripts

We considered several MiBench programs and an Ariane V core

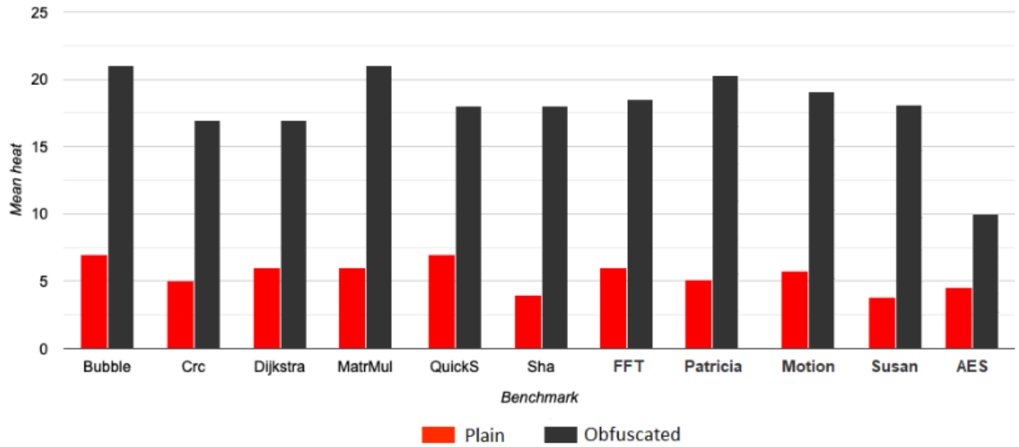
We measured:

- how many instructions are added
- the *register heat*

Program	#lines (Plain)	#lines (Protected)	Overhead
Bubble	109	189	73%
CRC	26	130	400%
Dijkstra	42	143	240%
MatrMul	99	255	158%
QuickS	115	224	95%
SHA	228	481	110%
FFT	42	194	361%
Patricia	869	1005	15%
Motion	507	659	30%
Susan	606	739	22%
AES	11419	12428	9%

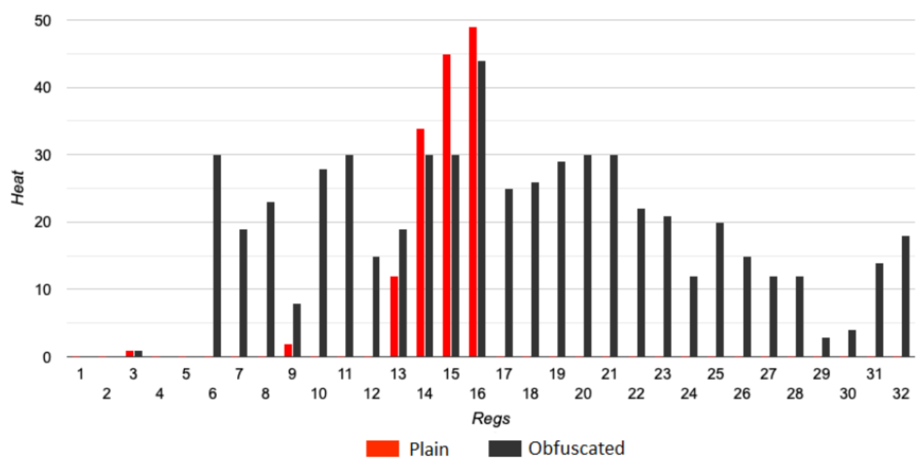
Hardware Trojan Horses: Design for Trust

Some results



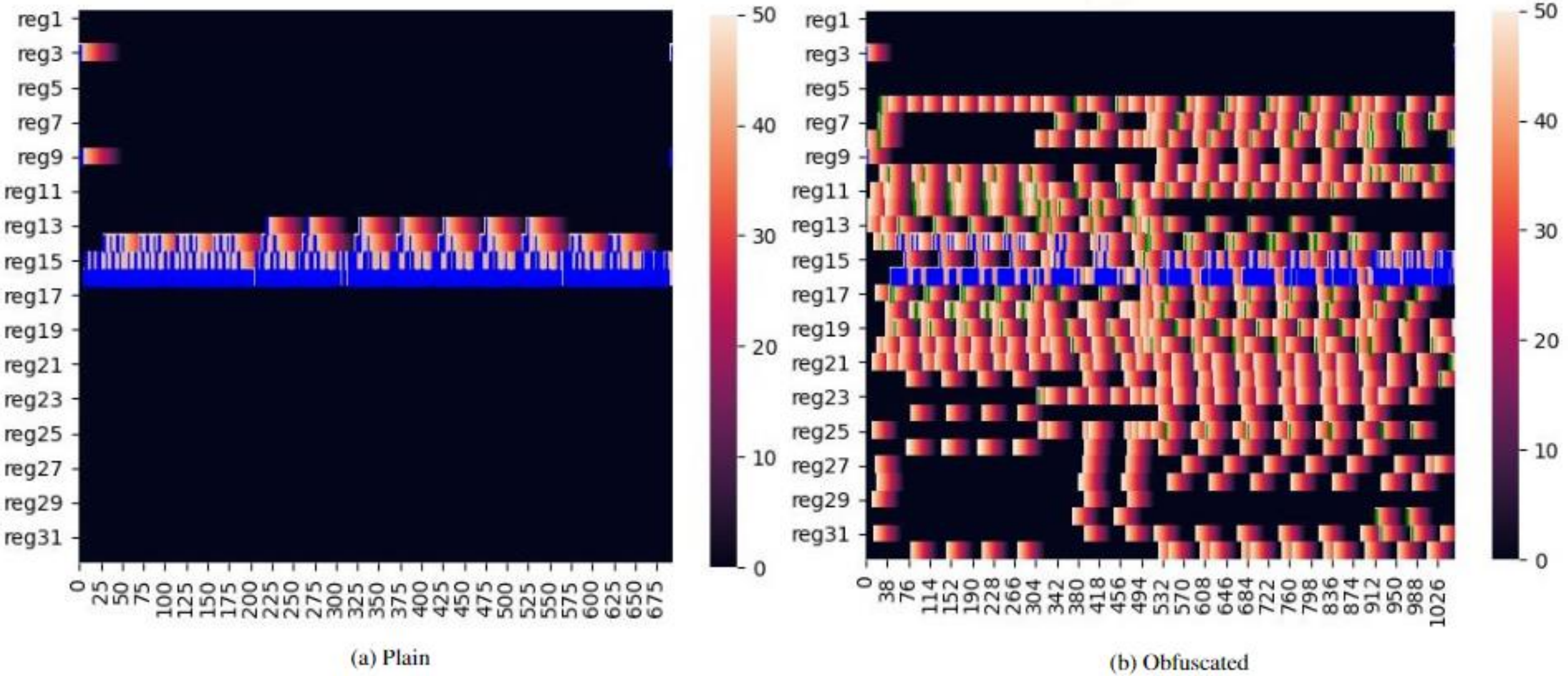
Per-Application Average Heat

Per-Register Average Heat for SHA



Hardware Trojan Horses: Design for Trust

Some results



Per-Instruction Cycle and Per-Register Heat for SHA

Hardware Trojan Horses: Design for Trust

Applied at:

- Bitstream generation

Considered attacker:

- CAD tool

Type of attack:

- Any Trojan

- Is Your FPGA Bitstream Hardware Trojan-free? Machine Learning Can Provide an Answer, Alessandro Palumbo et al., In Journal of Systems Architecture vol. 128, 2022

Hardware Trojan Horses: Design for Trust

Solution:

- Exploit Machine Learning to *compare* design-time and runtime features

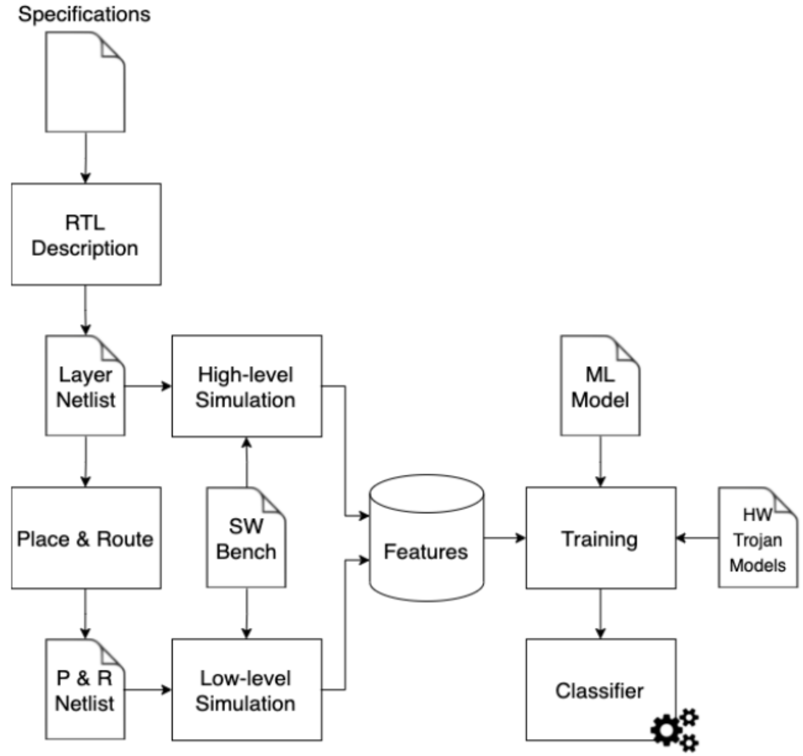
Limitations:

- Feasibility of accurate post-implementation features

Hardware Trojan Horses: Design for Trust

Training

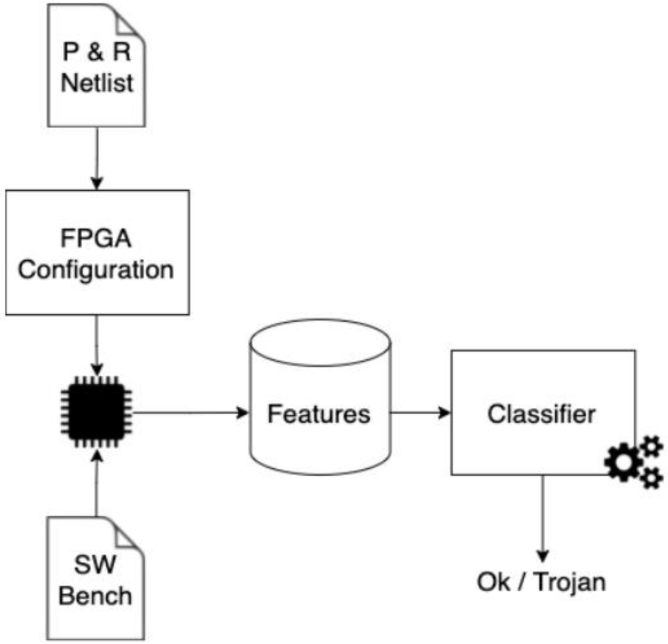
- Several features are extracted via netlist-level and post P&R-level simulations
  - Power consumption
  - Performance counters
  - …



Hardware Trojan Horses: Design for Trust

Testing

- The same features are then extracted via emulation on the target FPGA device
- Previously trained classifiers are employed to detect the presence of HTHs



Hardware Trojan Horses: Design for Trust

Some results

We implemented the framework using the R programming language

We analysed several ML engines:

- Random Forest
- Support Vector Machine
- Decision Tree
- Logistic Regression
- Gradient Boosting Machine

The dataset was composed of 60 good circuits and 80 infected circuits

We considered several MiBench programs and an Ibex core implemented onto a Xilinx XC7Z020 1CLG484C Zynq-7000 FPGA

Hardware Trojan Horses: Design for Trust

Some results

Benchmark	Description
Coremark	CPU performance benchmark
Median	Median image filter
Multiply	Numbers multiplication
Rsort	Sorting algorithm
Towers	Solver for the tower puzzle

Feature	Description
Benchmark	Program under execution
Cycles	# clock cycles to execute the program
InstrRet	# instructions retired in the program
LSUs	# waiting cycles to access data memory
FetchWait	# waiting cycles before instruction fetch
Load	# load instructions
Store	# store instructions
Jump	# jump instructions
CondBr	# conditional branches
TakCBran	# taken conditional branches
CompIns	# compressed
MulWait	# cycles for mul. completion
DivdWait	# cycles for div. completion
LUTs	# Look Up Tables
FFs	# Flip Flops
AvgDynPow	Avg. dyn. power consumption
AvgPower	Avg. total power consumption
Timing	Worst negative slack
Temperature	Temperature trend

Hardware Trojan Horses: Design for Trust

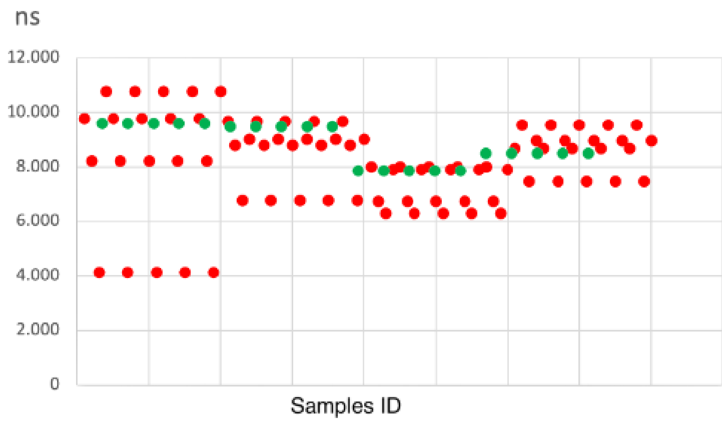
Some results

ML Model (65/35%)	Acc.
Dummy Classifier	0.5700
LR with Regularization	0.9368
SVM (Radial Basis Function)	0.9451
DT (algorithm C5.0)	0.9864
Random Forest	0.9872
Gradient Boosting Machine	0.9887

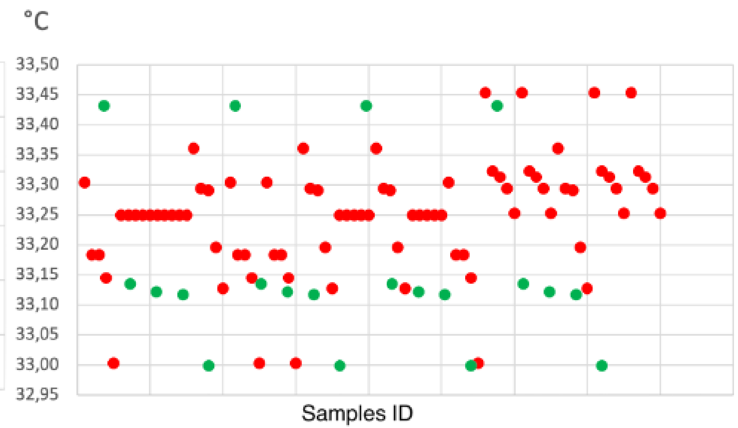
ML Model (80/20%)	Acc.
Dummy Classifier	0.5700
LR with Regularization	0.9578
SVM (Radial Basis Function)	0.9578
DT (algorithm C5.0)	0.9894
Random Forest	0.9921
Gradient Boosting Machine	0.9960

Hardware Trojan Horses: Design for Trust

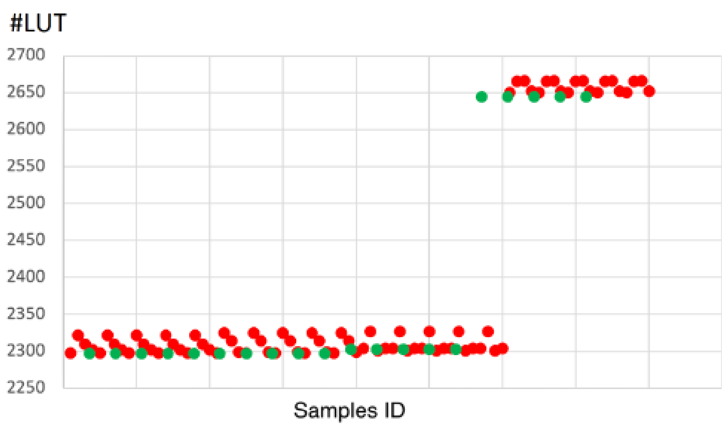
Some results



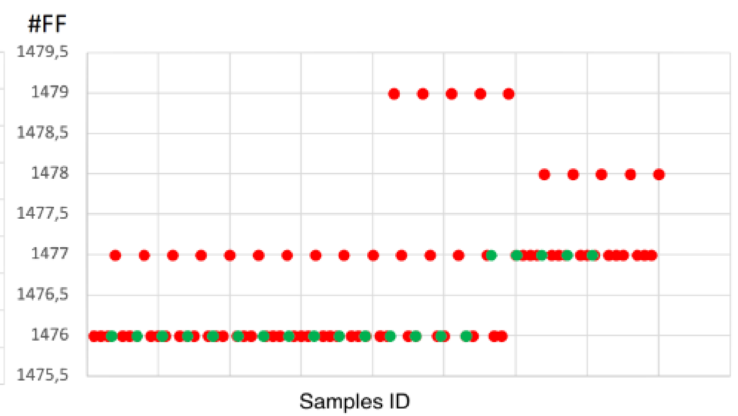
(a) Timing



(b) Temperature



(c) LUTs



(d) FFs

Students assignments

- S. Faezi, R. Yasaei, A. Barua and M. A. A. Faruque, "Brain-Inspired Golden Chip Free Hardware Trojan Detection," in IEEE Transactions on Information Forensics and Security, vol. 16, pp. 2697-2708, 2021
- T. Kurihara and N. Togawa, "Hardware-Trojan Classification based on the Structure of Trigger Circuits Utilizing Random Forests," 2021 IEEE 27th International Symposium on On-Line Testing and Robust System Design (IOLTS), Torino, Italy, 2022
- Q. Shi et al., "Golden Gates: A New Hybrid Approach for Rapid Hardware Trojan Detection using Testing and Imaging," 2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), McLean, VA, USA, 2019
- Z. Liu, J. Ye, X. Hu, H. Li, X. Li and Y. Hu, "Sequence Triggered Hardware Trojan in Neural Network Accelerator," 2020 IEEE 38th VLSI Test Symposium (VTS), San Diego, CA, USA, 2020
- V. R. Surabhi et al., "Hardware Trojan Detection Using Controlled Circuit Aging," in IEEE Access, vol. 8, pp. 77415-77434, 2020

References

- M. Tehranipoor, F. Koushanfar, A survey of hardware trojan taxonomy and detection, IEEE Desesign & Test of Computers (2010)
- T. Hoque, P. Slpsk, S. Bhunia, Trust issues in microelectronics: The concerns and the countermeasures, IEEE Consumer Electronics Magazine (2020)
- K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, M. Tehranipoor, Hardware Trojans: Lessons learned after one decade of research, ACM Transactions on Desisgn Automation of Electronic Systems 22 (2016) 6:1–6:23
- L. Lin, W. Burleson and C. Paar, "MOLES: Malicious off-chip leakage enabled by side-channels," 2009 IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers, San Jose, CA, 2009, pp. 117-122.
- https://www.spiegel.de/international/world/catalog-reveals-nsa-has-back-doors-for-numerous-devices-a-940994.html
- trust-hub.org
- Y. Liu, Y. Zhao, J. He, A. Liu and R. Xin, "SCCA: Side-channel correlation analysis for detecting hardware Trojan," 2017 11th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID), Xiamen, China, 2017, pp. 196-200.
- J. He, Y. Zhao, X. Guo and Y. Jin, "Hardware Trojan Detection Through Chip-Free Electromagnetic Side-Channel Statistical Analysis," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 10, pp. 2939-2948, Oct. 2017.
- J. Cruz, F. Farahmandi, A. Ahmed and P. Mishra, "Hardware Trojan Detection Using ATPG and Model Checking," 2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID), Pune, India, 2018, pp. 91-96
- H. Salmani and M. Tehranipoor, "Layout-Aware Switching Activity Localization to Enhance Hardware Trojan Detection," in IEEE Transactions on Information Forensics and Security, vol. 7, no. 1, pp. 76-87, Feb. 2012
- H. Salmani and M. Tehranipoor, "Analyzing circuit vulnerability to hardware Trojan insertion at the behavioral level," 2013 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS), New York, NY, USA, 2013, pp. 190-195
- F. S. Hossain, M. Shintani, M. Inoue and A. Orailoglu, "Variation-Aware Hardware Trojan Detection through Power Side-channel," 2018 IEEE International Test Conference (ITC), Phoenix, AZ, USA, 2018, pp. 1-10.
- M. Rathmair, F. Schupfer and C. Krieg, "Applied formal methods for hardware Trojan detection," 2014 IEEE International Symposium on Circuits and Systems (ISCAS), Melbourne, VIC, Australia, 2014, pp. 169-172
- E. Love, Y. Jin and Y. Makris, "Proof-Carrying Hardware Intellectual Property: A Pathway to Trusted Module Acquisition," in IEEE Transactions on Information Forensics and Security, vol. 7, no. 1, pp. 25-40, Feb. 2012
- Šišejković, Dominik, et al. "Control-lock: Securing processor cores against software-controlled hardware trojans." Proceedings of the 2019 on Great Lakes Symposium on VLSI. 2019.
- A. Basak, S. Bhunia, T. Tkacik and S. Ray, "Security Assurance for System-on-Chip Designs With Untrusted IPs," in IEEE Transactions on Information Forensics and Security, vol. 12, no. 7, pp. 1515-1528, July 2017
- C. Liu, J. Rajendran, C. Yang and R. Karri, "Shielding heterogeneous MPSoCs from untrustworthy 3PIPs through security-driven task scheduling," 2013 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS), New York, NY, USA, 2013, pp. 101-106
- J. J. Rajendran, O. Sinanoglu and R. Karri, "Building Trustworthy Systems Using Untrusted Components: A High-Level Synthesis Approach," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 24, no. 9, pp. 2946-2959, Sept. 2016
- A. Bolat, L. Cassano, P. Reviriego, O. Ergin and M. Ottavi, "A Microprocessor Protection Architecture against Hardware Trojans in Memories," 2020 15th Design & Technology of Integrated Systems in Nanoscale Era (DTIS), Marrakech, Morocco, 2020, pp. 1-6
- A. Palumbo, L. Cassano, P. Reviriego, G. Bianchi and M. Ottavi, "A Lightweight Security Checking Module to Protect Microprocessors against Hardware Trojan Horses," 2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Athens, Greece, 2021, pp. 1-6
- DETON: DEfeating hardware Trojan horses in microprocessors through software ObfuscatioN, Luca Cassano et al., In Journal of Systems Architecture, vol. 129, 2022
- On the optimization of Software Obfuscation against Hardware Trojans in Microprocessors, Luca Cassano et al., In DDECS 2022
- Is Your FPGA Bitstream Hardware Trojan-free? Machine Learning Can Provide an Answer, Alessandro Palumbo et al., In Journal of Systems Architecture vol. 128, 2022