Hardware Trojans in Processor Based Circuit: from Design to Countermeasures

David Hély, Laboratoire LCIS, Grenoble INP
david.hely@grenoble-inp.fr

GDR SoC-SiP – Journée « Sécurité des systèmes embarqués »
Contrefaçons, PUF et Trojans
Paris, le 27 novembre 2012

Grenoble INP

LABORATOIRE DE CONCEPTION ET D'INTEGRATION DES SYSTEMES

Outline

- Hardware Trojans
- Hardware Trojan in Processor based Design
  - Hardware Trojans Design Experiences
  - Information leakage Trojan
  - Instruction Modification Trojan
- Hardware Trojan Countermeasures
  - Detection Methods overview
  - Run time Detection: the PPU example
- Conclusions

Hardware Trojans: Definitions

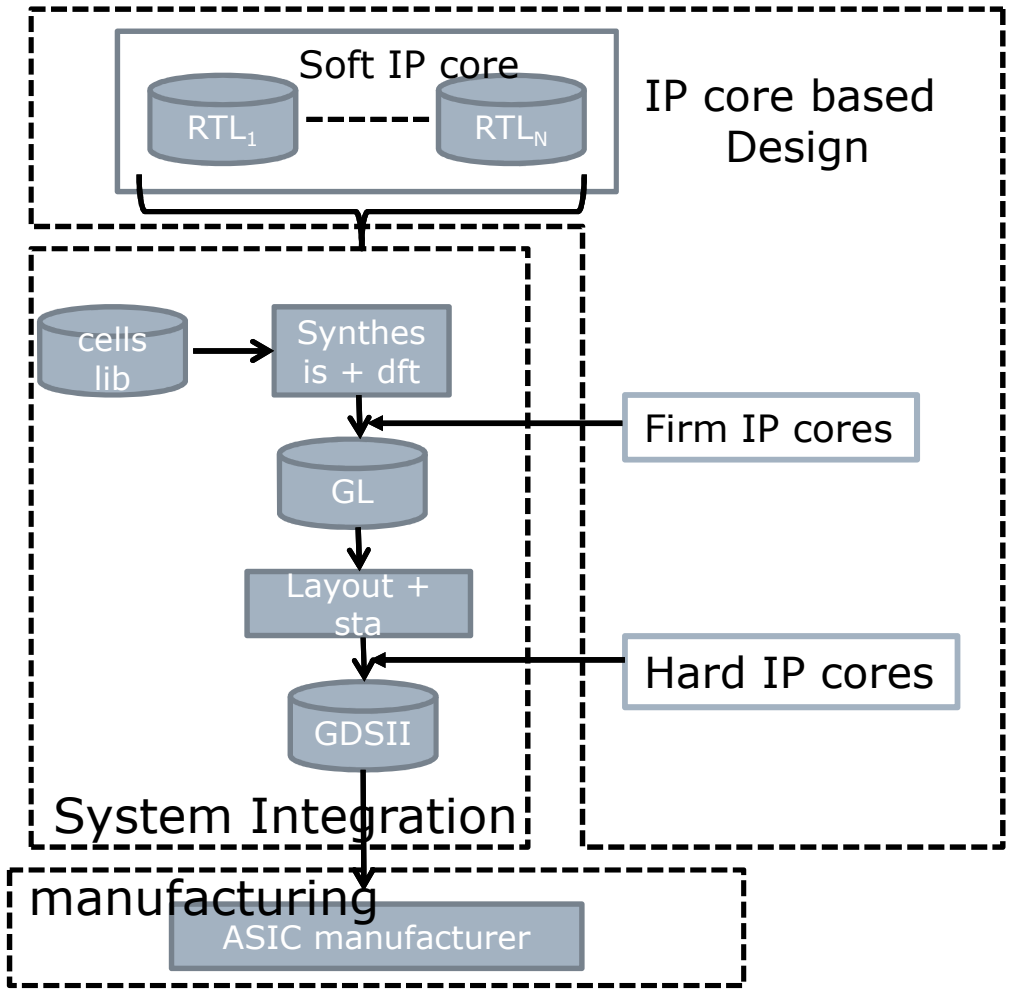
- Some definitions:
  - Hardware Trojan : malicious modification of an integrated circuit in order to offer advantage to an adversary
  - Trigger: Activation mechanism of the Trojan
  - Payload: Effect of the modification (deny of service, secret leakage…)

Insertion Scenario

- When and by who a Hardware Trojan can be inserted?
  - Different tasks of the design process
    - From Architecture to fabrication
  - Different level of abstraction
    - RTL, gate, layout
  - Different stake-holders
    - IP Providers, Fabrication facilities

Karri et Al « Trustworthy Hardware : Identifying and Classifying Hardware Trojans » IEEE Computer 2010

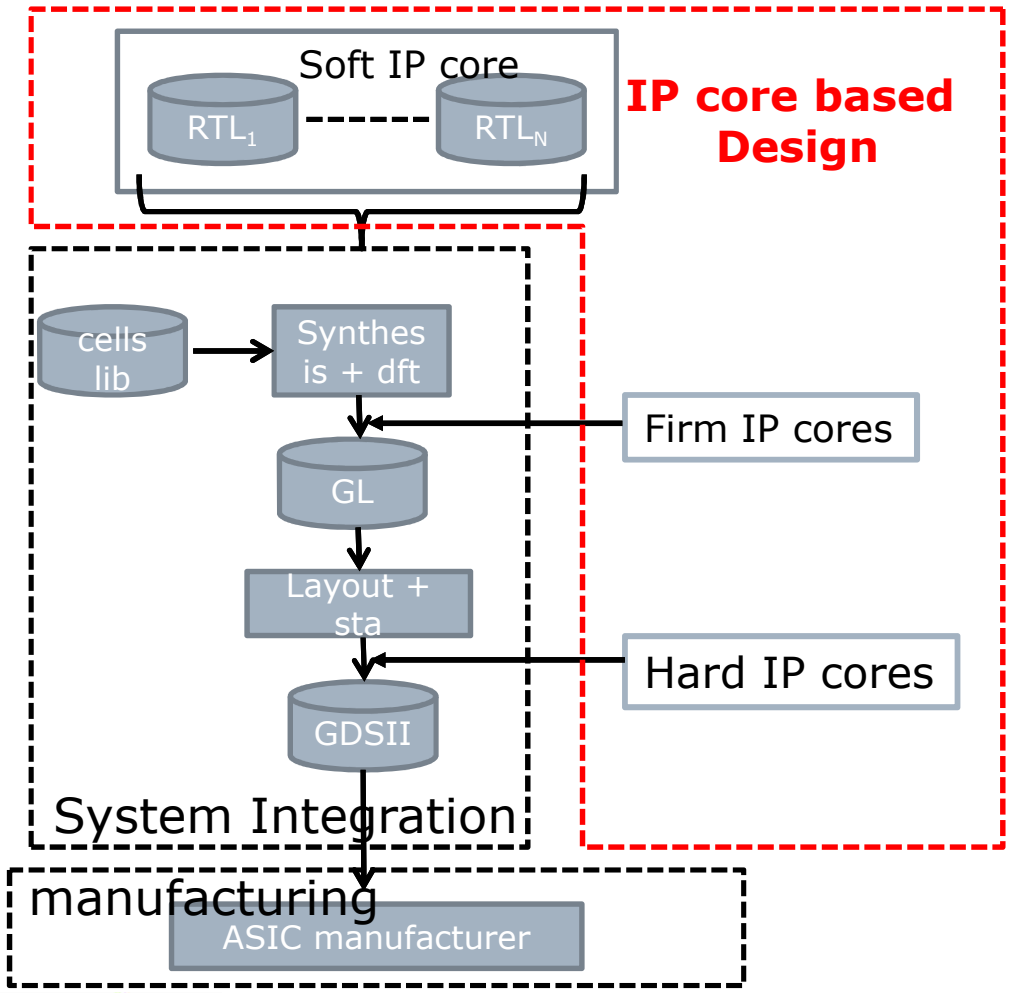
Implanting Stage



- **When can be the Trojan implanted?**
- **By who?**

Credit: Hardware Trojan Detection Solutions and Design-for-Trust Challenges, Tehranipoor et al.

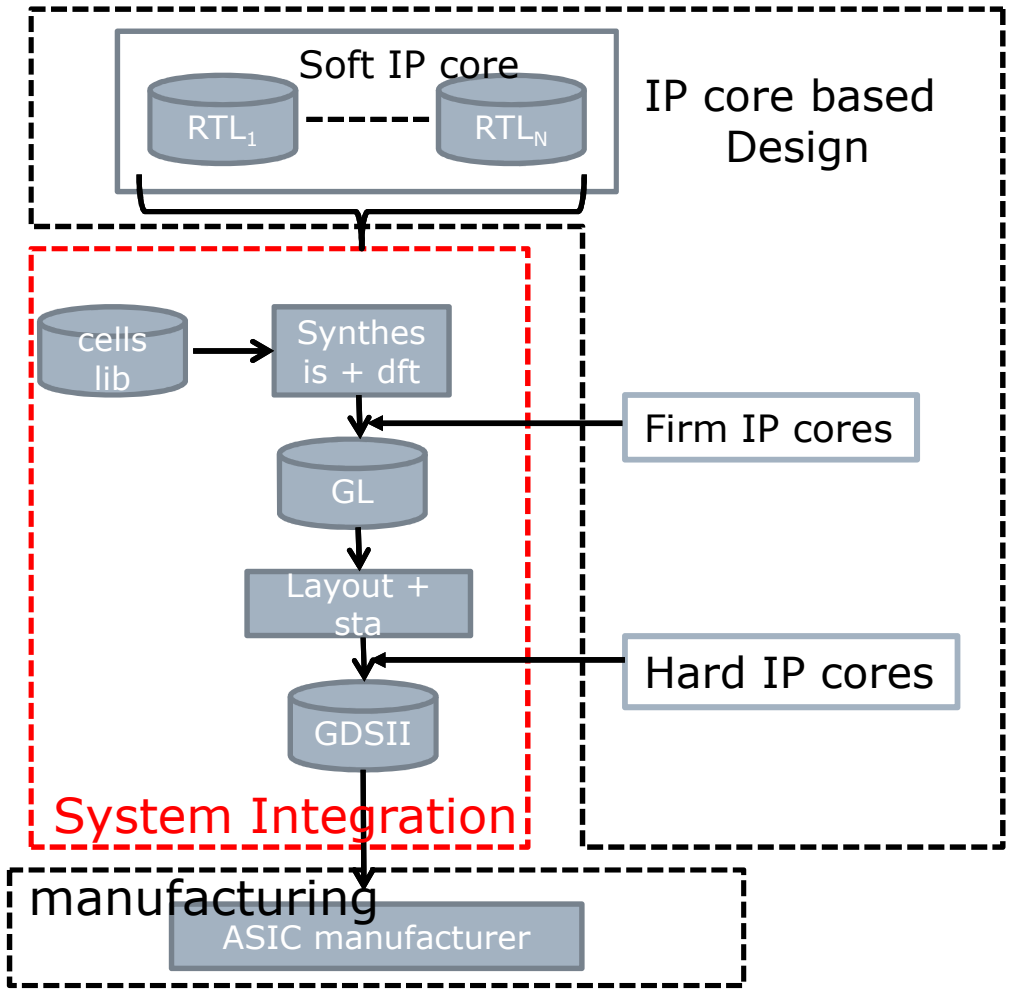
Implanting stage



- **When can be the Trojan implanted?**
  - **Within an IP**
- **By who?**
  - **IP provider**
  - **Malicious Designer**

Credit: Hardware Trojan Detection Solutions and Design-for-Trust Challenges, Tehranipoor et al.

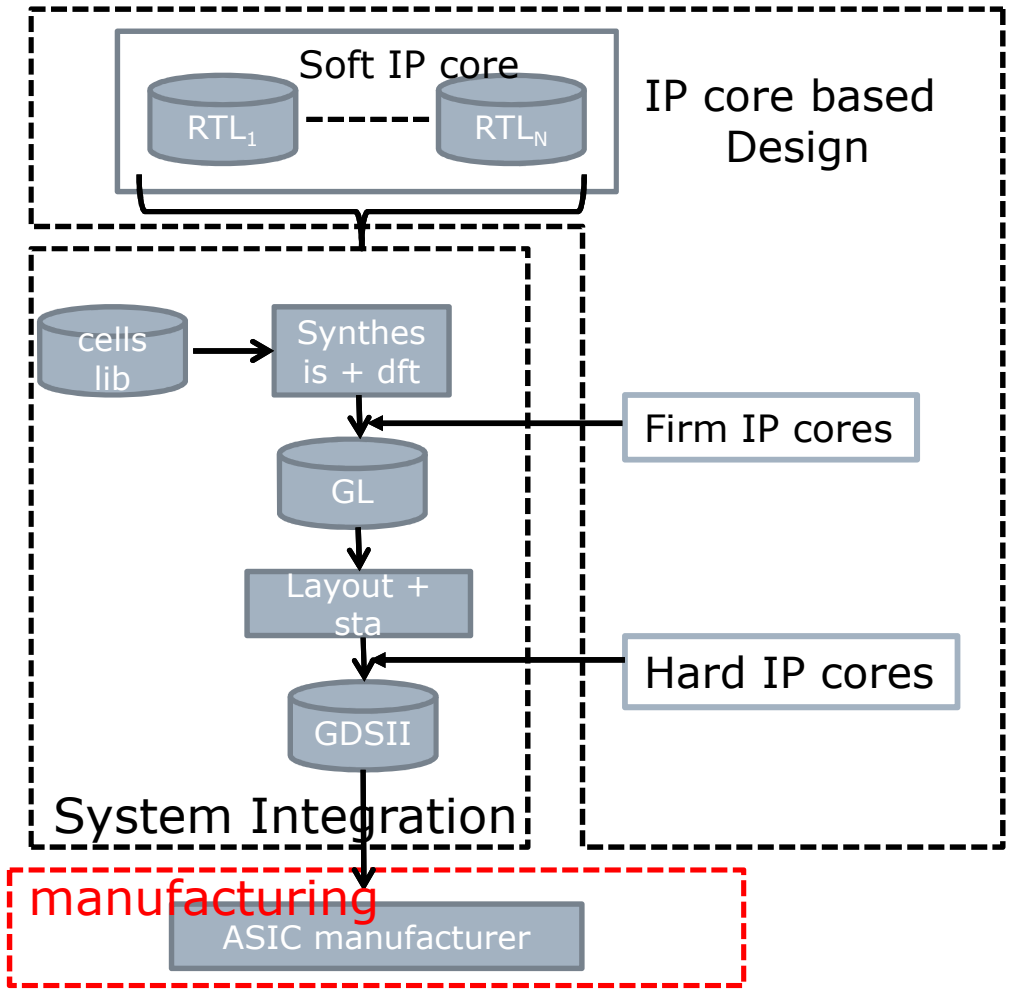
Implanting stage



- **When can be the Trojan implanted?**
  - **During the design flaw**
- **By who?**
  - **Malicious tool**
  - **Malicious System Designer**

Credit: Hardware Trojan Detection Solutions and Design-for-Trust Challenges, Tehranipoor et al.

Implanting stage



- **When can be the Trojan implanted?**
  - **During the fabrication**
- **By who?**
  - **Test Engineer**
  - **Process engineer**

Credit: Hardware Trojan Detection Solutions and Design-for-Trust Challenges, Tehranipoor et al.

Outline

- Hardware Trojans
- Hardware Trojan in Processor based Design
  - Hardware Trojans Design Experiences
  - Information leakage Trojan
  - Instruction Modification Trojan
- Hardware Trojan Countermeasures
  - Detection Methods overview
  - Run time Detection: the PPU example
- Conclusions

Experiences in Hardware Trojan Design

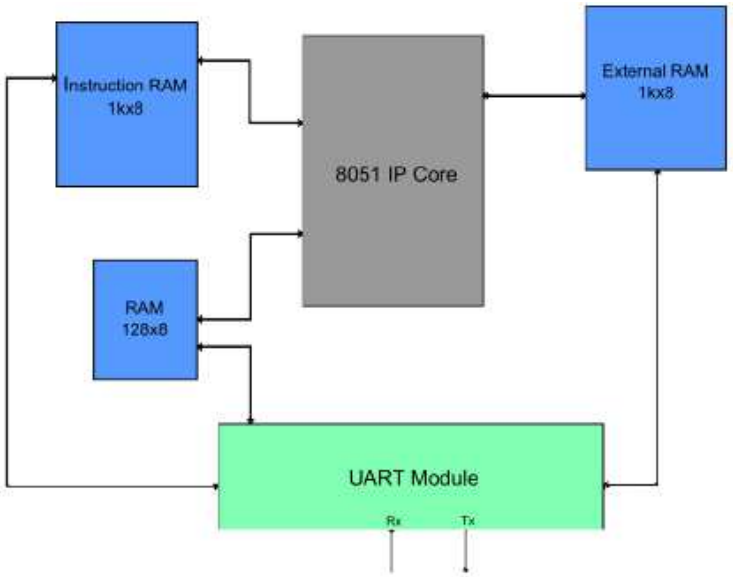
- CSAW 2011: Embedded Systems Challenge

NYU·poly
POLYTECHNIC INSTITUTE OF NYU

- Student Competition focusing on hardware Trojan Design within an 8051 based circuit running RC5 encryption

Hardware Trojan in a 8051

- Scenario:
  - Full control on the HDL
    - Malicious Designer…

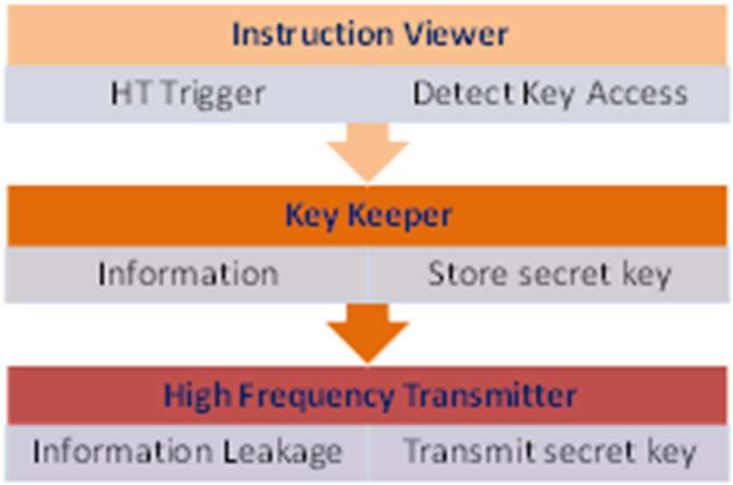


- Purposes of the design modifications
  - Leak secret information via side channel
  - Denial of service
  - Untrusted software computing

"Key Emission via Hardware Trojan", ICCD 2012

Information Leakage Trojan

- Detect any RC5 encryption starts
- Store the data under attack
- Leak secret information (i.e. key) via the communication line



The Trigger

- Purpose: Detect any RC5 encryption
  - Based on the RC5 algorithm

$$
\begin{aligned}
&A = A + S[0];\\
&B = B + S[1];\\
&\textbf{for } i = 1 \textbf{ to } r \textbf{ do}\\
&\qquad A = ((A \oplus B) \lll B) + S[2 * i];\\
&\qquad B = ((B \oplus A) \lll A) + S[2 * i + 1];
\end{aligned}
$$

  - Detect access to the extended key (pattern matching)
    - Look for instruction byte corresponding to a move followed by an add.

Pros	Cons
Low cost design	Specific to the platform architecture Risk of false positive

Storing the secret information

- Goal: storing the extended key
- Several options:
  - Dedicated registers
  - Reuse of processor memory

Storing the secret information

- Purpose: storing the extended key
- Several options:
  - **Dedicated registers**
  - Reuse of processor memory

Pros	Cons
Low cost circuitry control	Additional Flip-flops

Storing the secret information

- Purpose: storing the extended key
- Several options:
  - Dedicated registers
  - **Reuse of processor memory**

Pros	Cons
Low area overhead	Control circuitry

Data Transmission

- Principle: Reuse of the serial link[1]
- Constraints:
  - Data reconstruction must be easy
  - No perturbation of the original link
- Options
  - Encode data using protocol parameters[2]
  - Add a high frequency signal

[1] Baumgarten et al "Case Study in Hardware Trojan Design and Implementation", International Journal of Information Security 2011

[2] Jin et al, "Hardware Trojans in wireless cryptographic integrated circuits", Design and Test of Computers 2009

Data Transmission

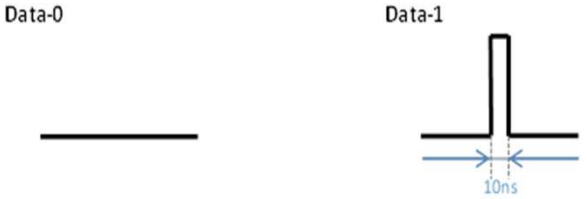
- Principle: Reuse of the serial link[1]
- Constraints:
  - Data reconstruction must be easy
  - No perturbation of the original link
- Options
  - Encode data using signal parameters[2]
  - **Add a high frequency signal**

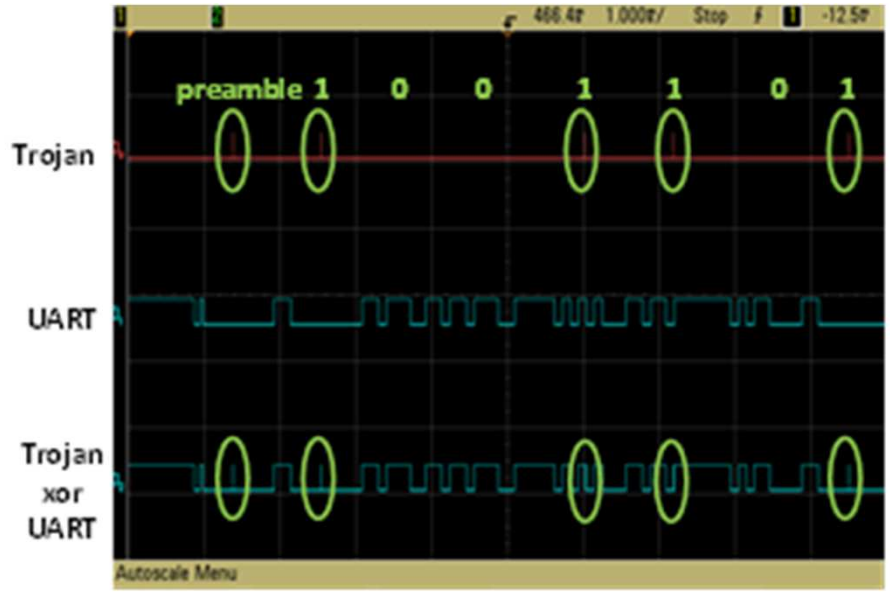
[1] Baumgarten et al "Case Study in Hardware Trojan Design and Implementation", International Journal of Information Security 2011

[2] Jin et al, "Hardware trojans in wireless cryptographic integrated circuits", Design and Test of Computers 2009

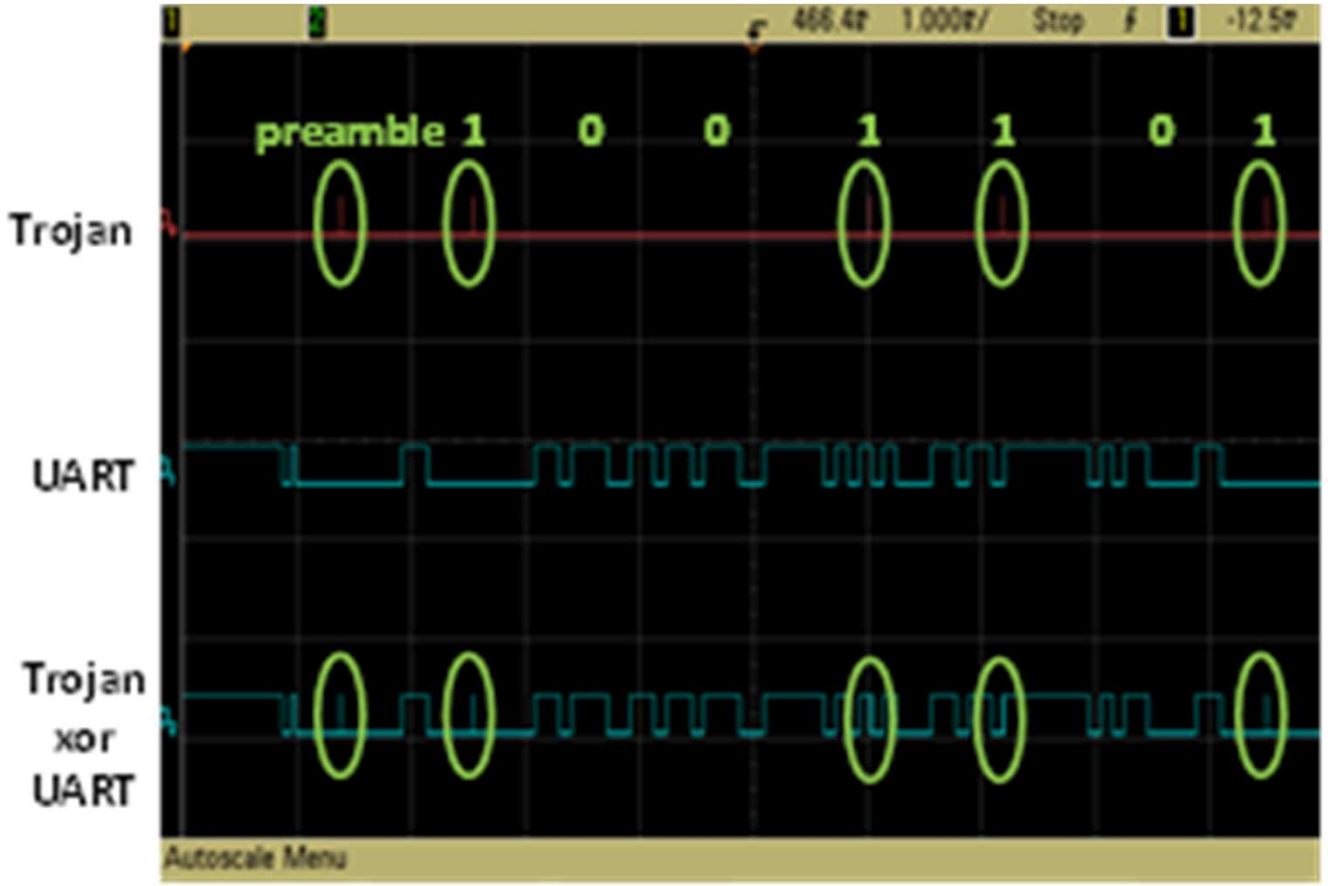
Data Transmission

- Data encoding and Preamble

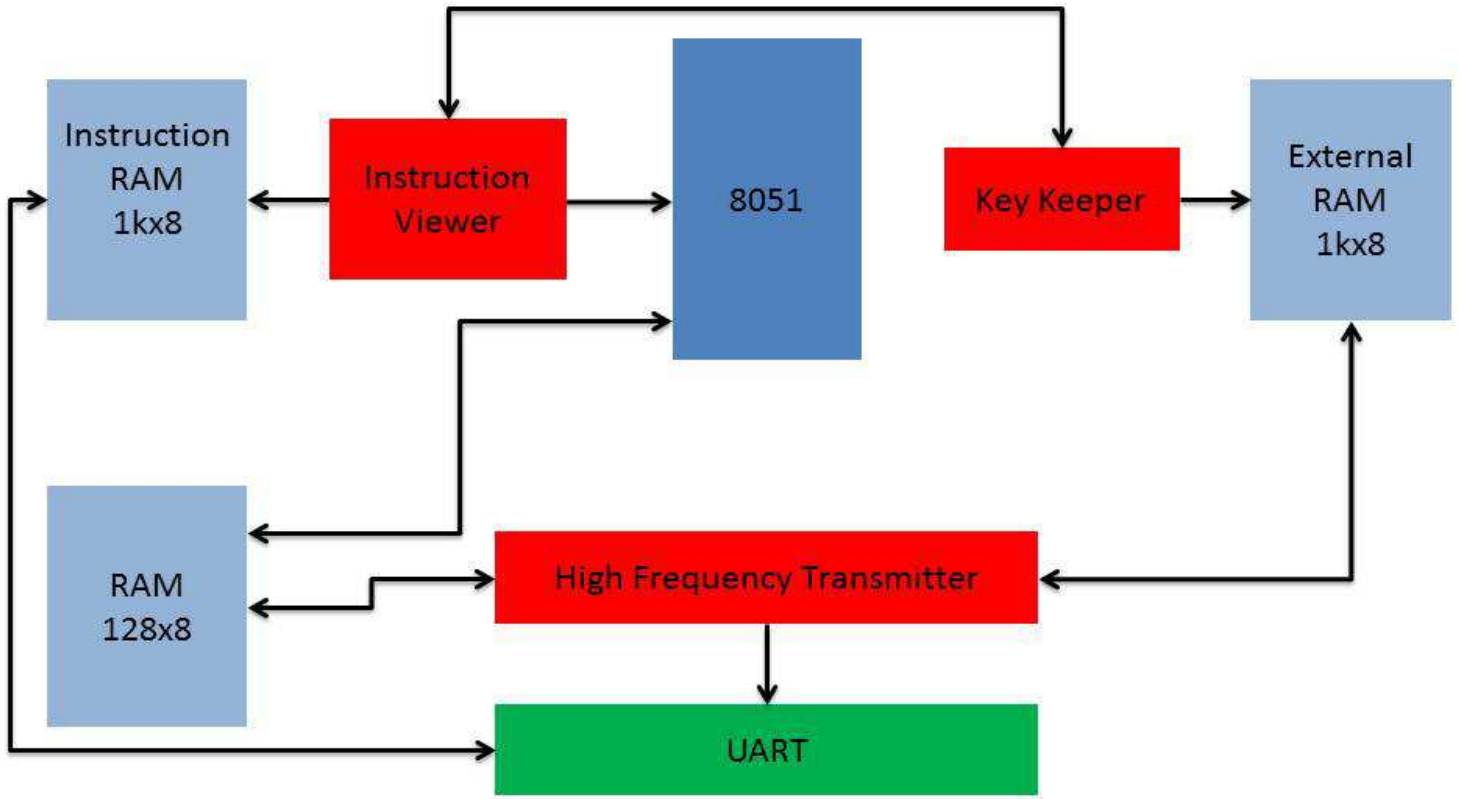




Data Transmission



Hardware Trojan Architecture



Area overhead: 2%

Side Channel Trojan

- Information leakage without degradations of the system performance
  - Information is hidden in the original signal
- Functionality of the digital part is not changed
- Processor based design offers many trigger opportunity based on instructions sequence (pattern matching)
- Defense against such Trojan:
  - Advanced statistical analysis of the transmission signal may reveal the Trojan

Jin et al, "Hardware Trojans in wireless cryptographic integrated circuits", Design and Test of Computers 2009

Instruction Set Modification

- Modification of the instruction set:
  - Malicious code execution via
    - Instruction replacement
    - Unused instruction
    - Instruction modification
    - Interrupt routing

Instruction Modification

- LCALL -> ACALL transformation
- Minimum modification in the rtl description

```
IC_ACALL when s_command = LCALL and (trojan_en='1')
IC_LCALL when s_command = LCALL and (trojan_en='0')
```

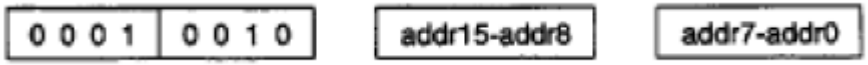
And

```
If (prev_instruction = RET) then trojan_en <='0';
else
trojan_ <='1';
end if;
```


Jin et al « Exposing Vulnerabilities of Untrusted Computing Platform », ICCD 2012

Instruction Modification

- Attacking scenario:
  - Decrement stored PC in the stack (ACALL is a two byte instruction)
  - Execute the malicious code at the jump of the ACALL
  - LCALL encoding



  - ACALL encoding

  - The jump address is PC<15:11>addr15-addr8

Jin et al « Exposing Vulnerabilities of Untrusted Computing Platform », ICCD 2012

Instruction Modification

- LCALL->ACALL
  - When the LCALL is fetched:
    1. ACALL is executed
    2. Malicious code is called
    3. LCALL is executed
  - Software runs its full original sequence
  - The Trojan payload can be very complex
  - Timing performance overhead

Outline

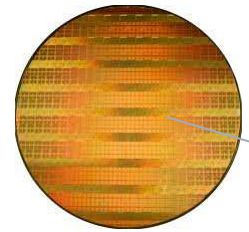
- Hardware Trojans
- Hardware Trojan in Processor based Design
  - Hardware Trojans Design Experiences
  - Information leakage Trojan
  - Instruction Modification Trojan
- Hardware Trojan Countermeasures
  - Detection Methods overview
  - Run time Detection: the PPU example
- Conclusions

Securing the Design Against HT

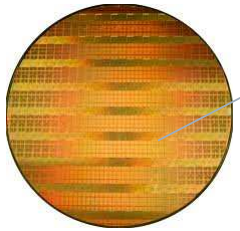
- Still the same question
  - Against who?
    - Untrusted IP, Untrusted designer
    - Untrusted fab
  - Countermeasures are then different according the availability of a golden model:
    - Check the design against a golden model
    - Try to find an “out of spec” behavior via testing
    - Use runtime check to prevent “out of spec” behavior

Detection via Side Chanel Analysis

- Purpose: compare side channel signature of the circuit against the golden model



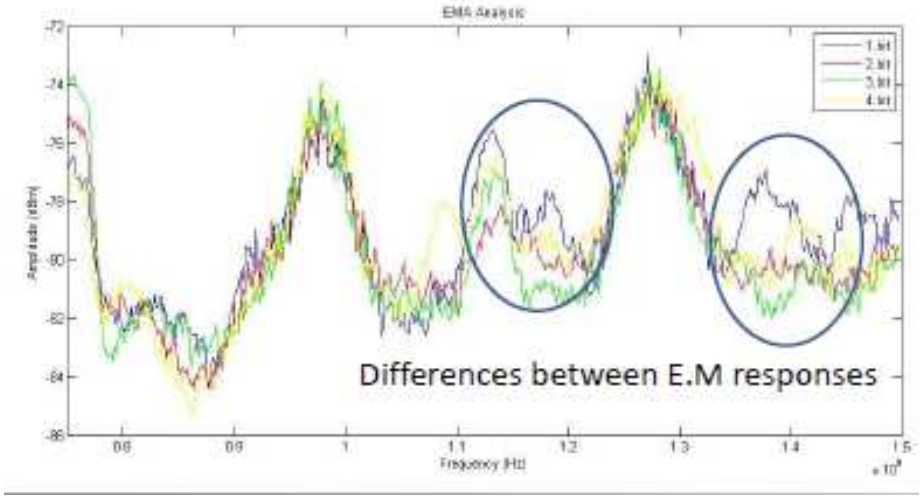
Golden chip



Chip under test

Side Channel Measurement

EM, power, Delay



Piroux et al "Trojan Detection via EM Analsysis" Esisar Team CSAW 2012

Functionnal Testing

- A new set of pattern is added to the test set in addition to the defect testing patterns:
  - Trojan testing patterns targeting hardware Trojan
  - *May be equivalent to looking for a needle into a haystack due to the large input space*
- Trojan Vectors (Wolf et al 2008)
  - Identify rare events and generate test vectors that trigger them
- Dummy flip-flops (Salmani et al 2009)
  - Increase the probability of rarely activate events by insertion of dummy flip-flops.

Run Time Protections

- Side Channel Analysis requires to have a golden model
- Other methods detection requires to activate the Trojan
  - Difficult for processor based design, the trigger may depend of a sequence of instructions
- Run-Time protection does not care of trigging the Trojan
  - Must just be able to detect abnormal behavior
  - Analog to safety

The Processor Protection Unit*

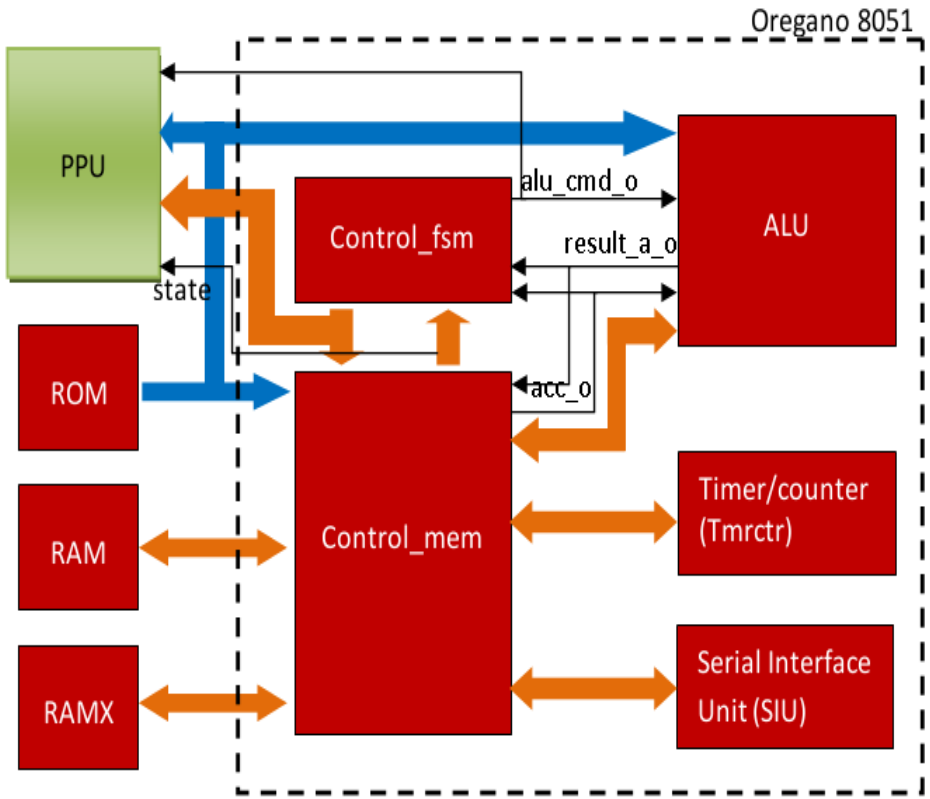
- Motivations:
  - Trusted IP should
    - monitor untrusted IP
    - be resilient to HT insertion
  - Processor monitoring against smart Trojans modifying the CPU behavior
  - Does not protect against DoS or side channel Trojans

*Master Thesis of Jérémy Dubeuf (Poly NY, Grenoble INP)

Processor Protection Unit

The PPU verifies:

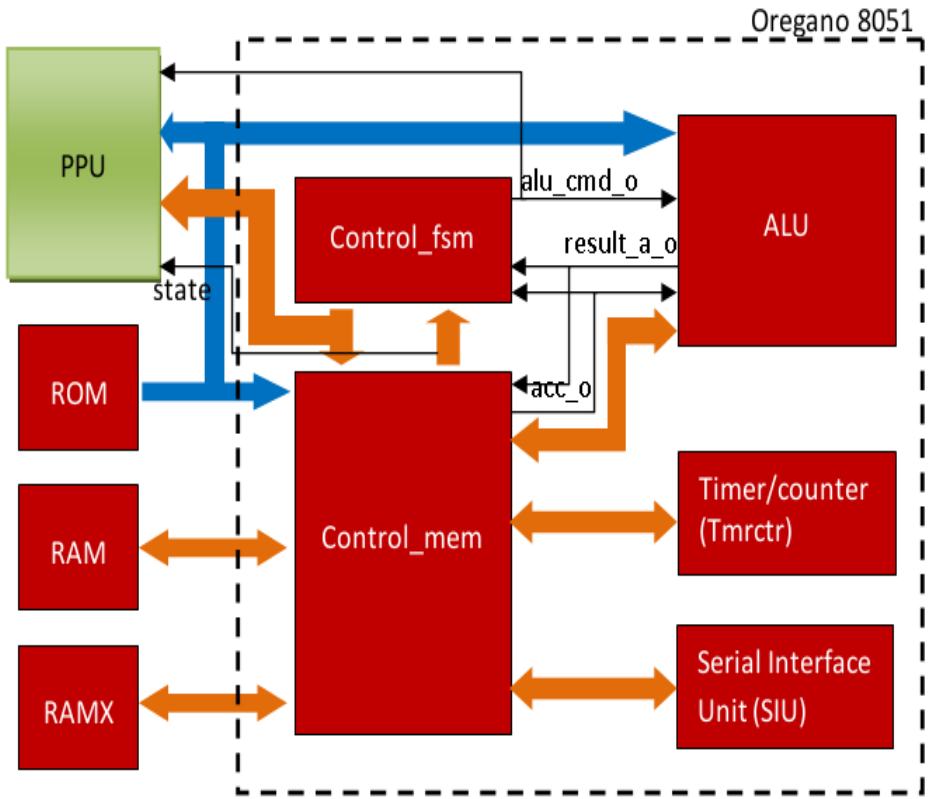
- Opcode value
- Instructions cycles
- FSM sequence
- Processor internal signals



Processor Protection Unit

The PPU verifies:

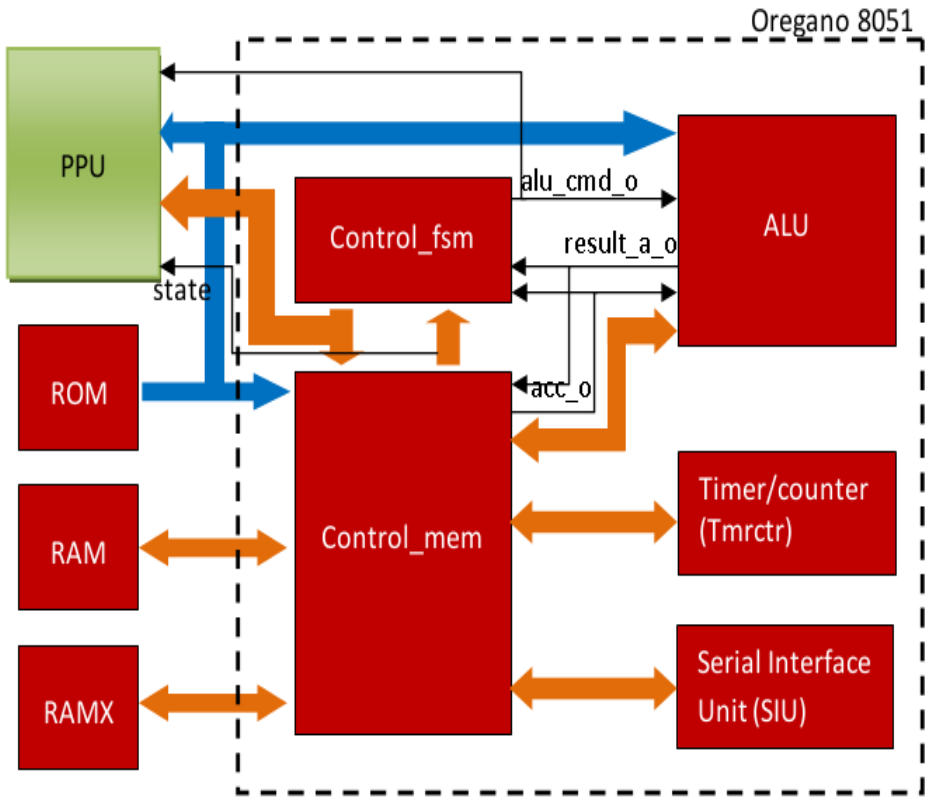
- Opcode value:
  - Against instruction insertion



Processor Protection Unit

The PPU verifies:

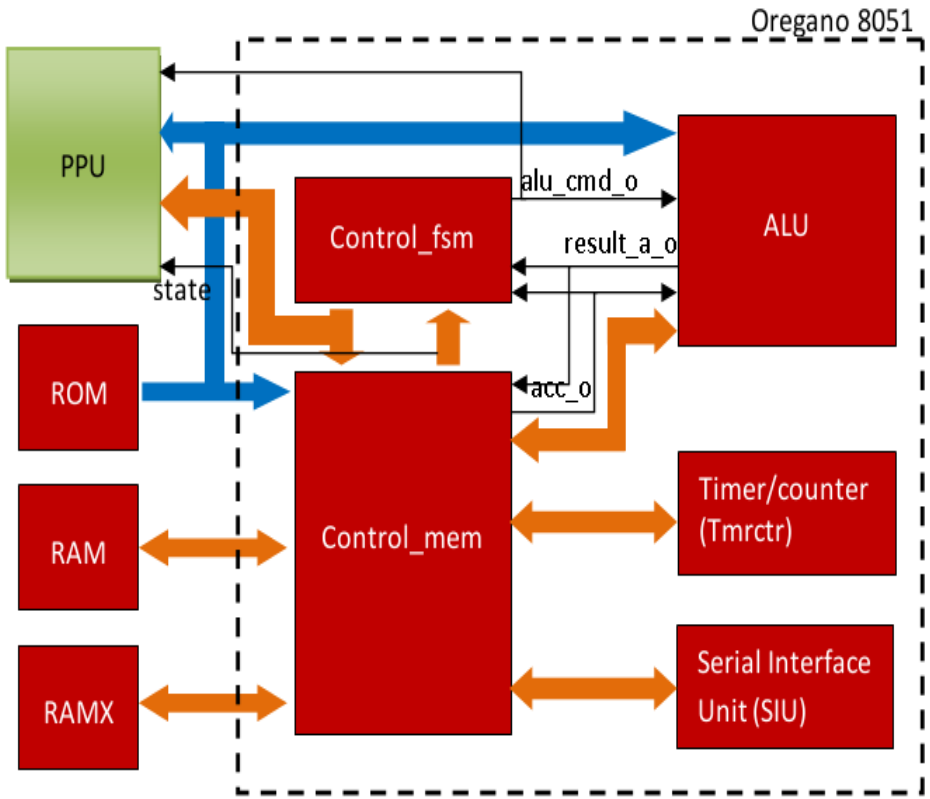
- # cycles per instruction:
  - Instruction modification



Processor Protection Unit

The PPU verifies:

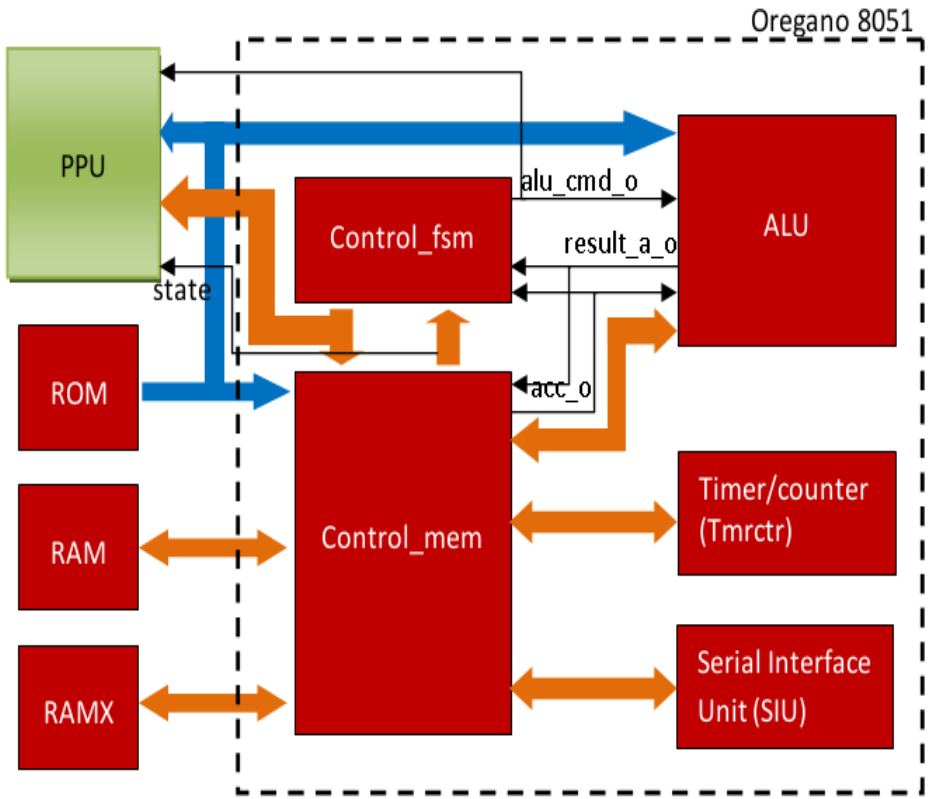
- FSM sequence
  - # of cycles does not change but the operation does



Processor Protection Unit

The PPU verifies:

- Internal signals
  - No extra operations



Trust in the PPU

- What about a Trojan within the PPU?
  1. The design must be as simple as possible to increase its testability
  2. The architecture must be resilient against malicious modifications

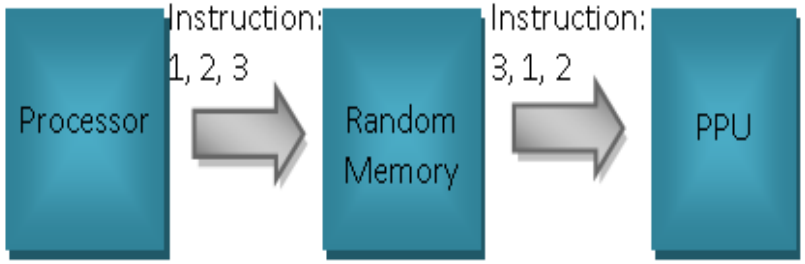
Trojan in the PPU

- A Trojan always active in the PPU may be detected by functional validation since the PPU is a table
- The hazards come from Trojan trigged by a sequence of instructions
  - The Trojan is in both the PPU and the processor

  => The PPU must break the software sequences

Hardenning the PPU

- Breaking the sequence
  - Instructions are stored in a Random memory



  - The information are stored in a random order
  - Information stored:
    - Instruction+ processor data

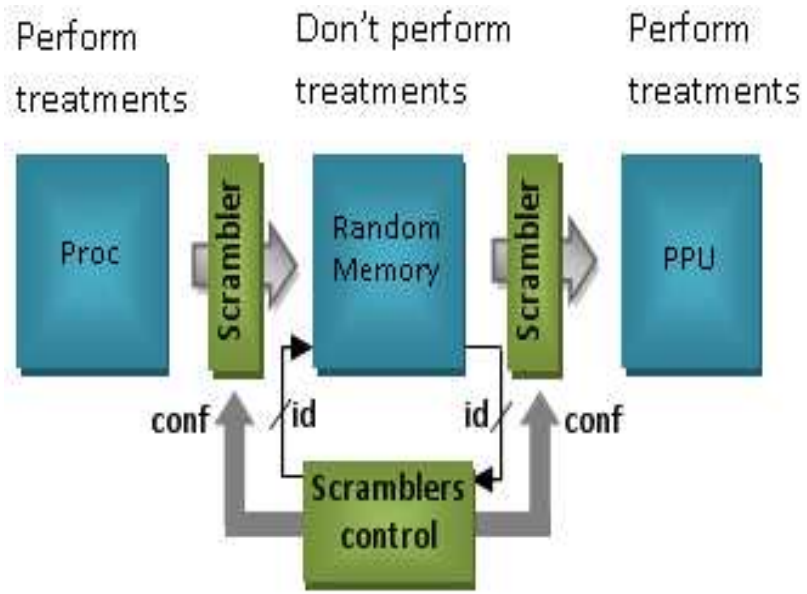
Hardenning the PPU

- What about a HT within the random memory?
  - Data in the memory can be corrupted:
    - Replacing illegal data by legal ones
    - Removing illegal activities

Data stored in memory must be hardly exploitable

Hardenning the PPU

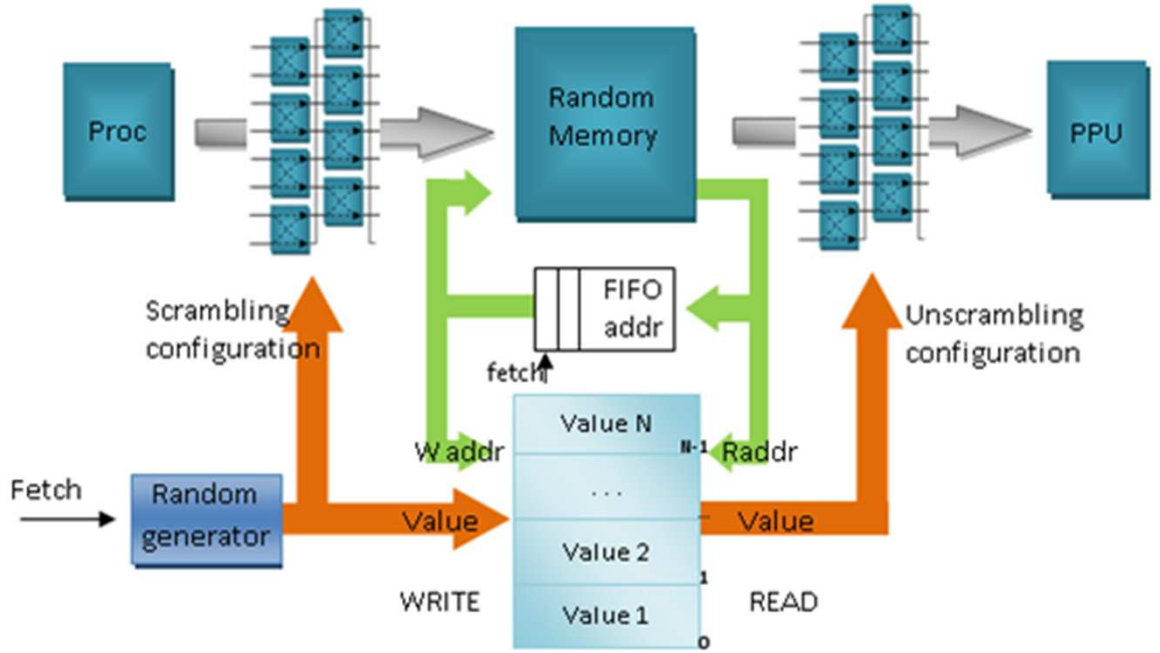
Data Scrambling



A trojan inside the random memory can hardly exploit the data

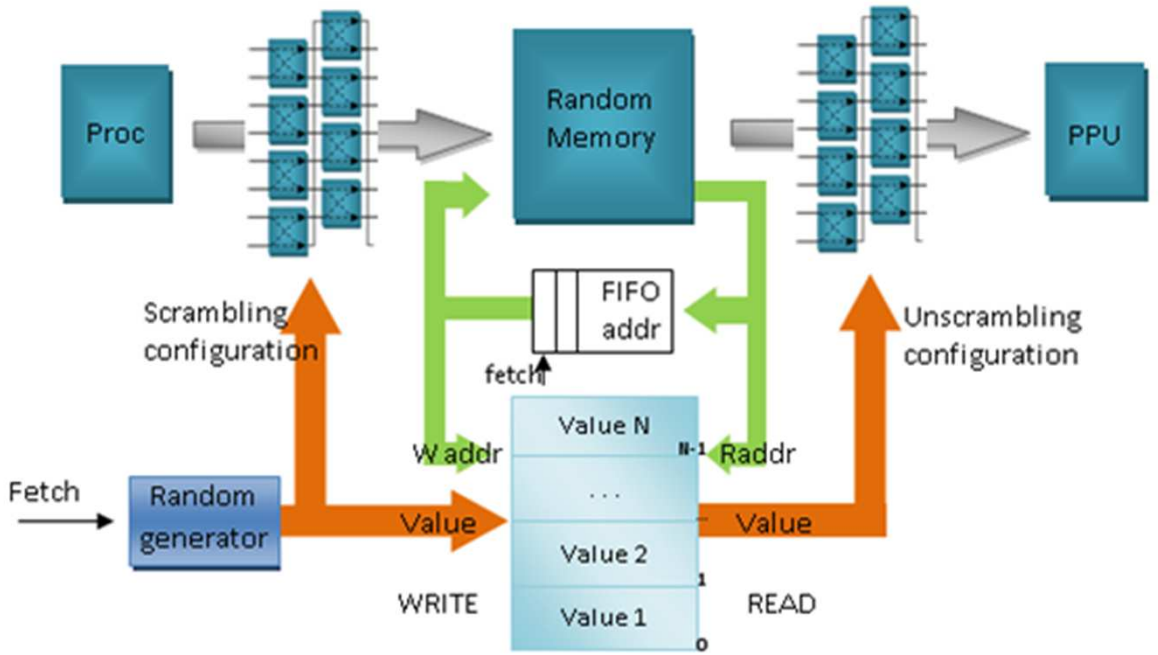
PPU Final Architecture

- Verification space is reduced to:
  - The scrambler
  - The random address generation
  - The interfaces between the PPU and the CPU



PPU Final Architecture

- Results:
  - All the CSAW 2011 Trojans are detected
  - The area overhead is 15 per cent of the CPU



Outline

- Hardware Trojans
- Hardware Trojan in Processor based Design
  - Hardware Trojans Design Experiences
  - Information leakage Trojan
  - Instruction Modification Trojan
- Hardware Trojan Countermeasures
  - Detection Methods overview
  - Run time Detection: the PPU example
- Conclusions

Conclusions

- Processor based circuit are vulnerable
  - Computing offers many possibilities for:
    - Smart trigger
    - Complex payload
- IP based design is a potential threat
  - No golden model
  - Run time detection may be a solution
- On going work
  - PPU enhancement
  - Design of specified Hardware Trojan
    - Hardware Trojan in RFID IC

Hardware Trojans in Processor Based Circuit: from Design to Countermeasures

David Hély, Laboratoire LCIS, Grenoble INP
david.hely@grenoble-inp.fr

GDR SoC-SiP – Journée « Sécurité des systèmes embarqués »
Contrefaçons, PUF et Trojans
Paris, le 27 novembre 2012

Grenoble INP

LABORATOIRE DE CONCEPTION ET D'INTEGRATION DES SYSTEMES