Hardware Trojans: A Threat for CyberSecurity

Julien Francq
julien.francq@cassidian.com

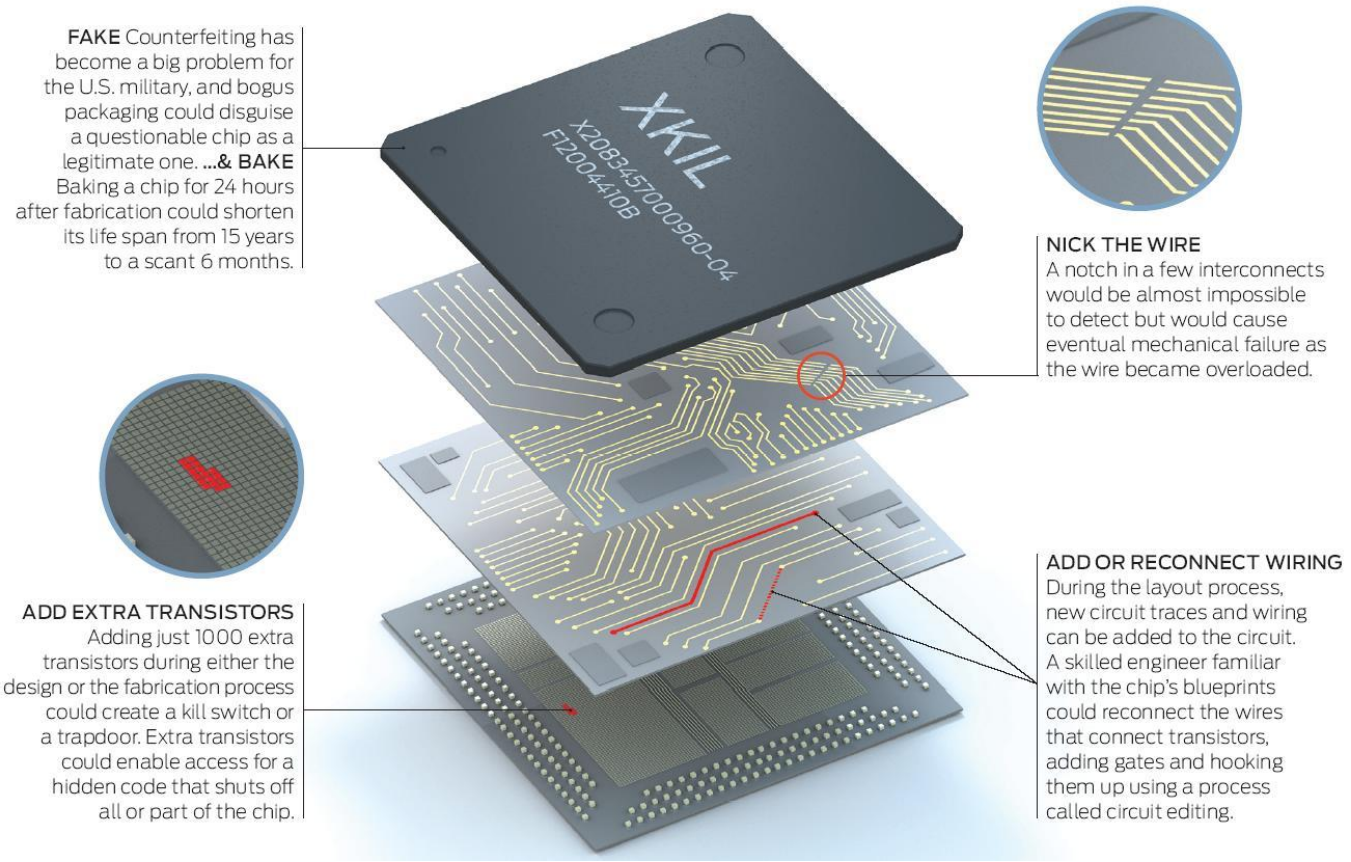
Cassidian CyberSecurity

2013, July the 8th

Outline

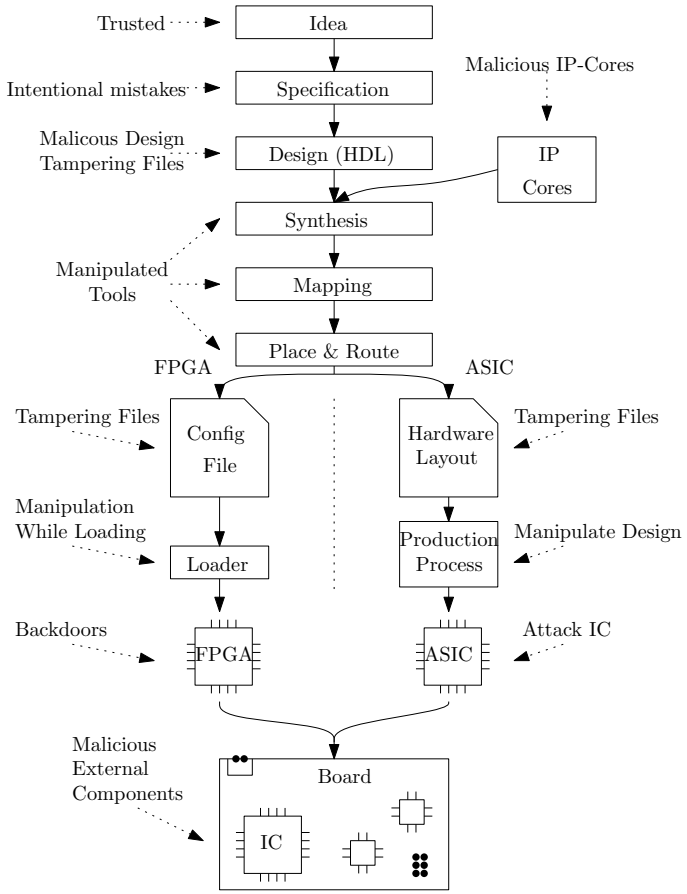
Hardware Trojan (HT)

- Malicious modifications of an Integrated Circuit (IC) during its design flow



Context

- Outsourcing of the fabrication of the ICs
- Difficult to ensure the trust in all the steps of the design flow



Hardware Trojans in Practice

- 2005 : US Department of Defense
- 2007 : DARPA "Trust in IC Program"
- 2007 : Israël *vs.* Syria
- 2009 : "Hot Topic" of CHES conference
- After 2009 : other conferences (DATE, HOST, CARDIS, ReConFig, *etc.*)
- [Skorobogatov *et al.* : "Breaktrough Silicon Scanning Discovers Backdoor in Military Chip", CHES 2012]
- $\Rightarrow$ HTs : real and emerging threat

Quantification of Risks

	Overproduction	Software cloning	HTs
Attackers	Fab	Competitors	Terrorists
Goal	Feed the Grey Market	IP Theft	Denial of Service, Data Theft, Sabotage
Impact	Economical	Economical	Risks on Security, Economy, Infrastructures (Society)
Risks	+++	++	+

- Impact $\times$ Risks too important to be neglicted

Possible Payloads

- Kill switch
  - Fighters
- Dysfonctional circuit
  - Satellite which works only 6 months
- Secret information leakage
  - Ciphered communications
- Help a malware by providing a backdoor
  - Privilege escalation, automatic login, password theft
- Prevent from going to sleep mode
  - Autonomy
- *etc.*

Hardware Trojan Taxonomy

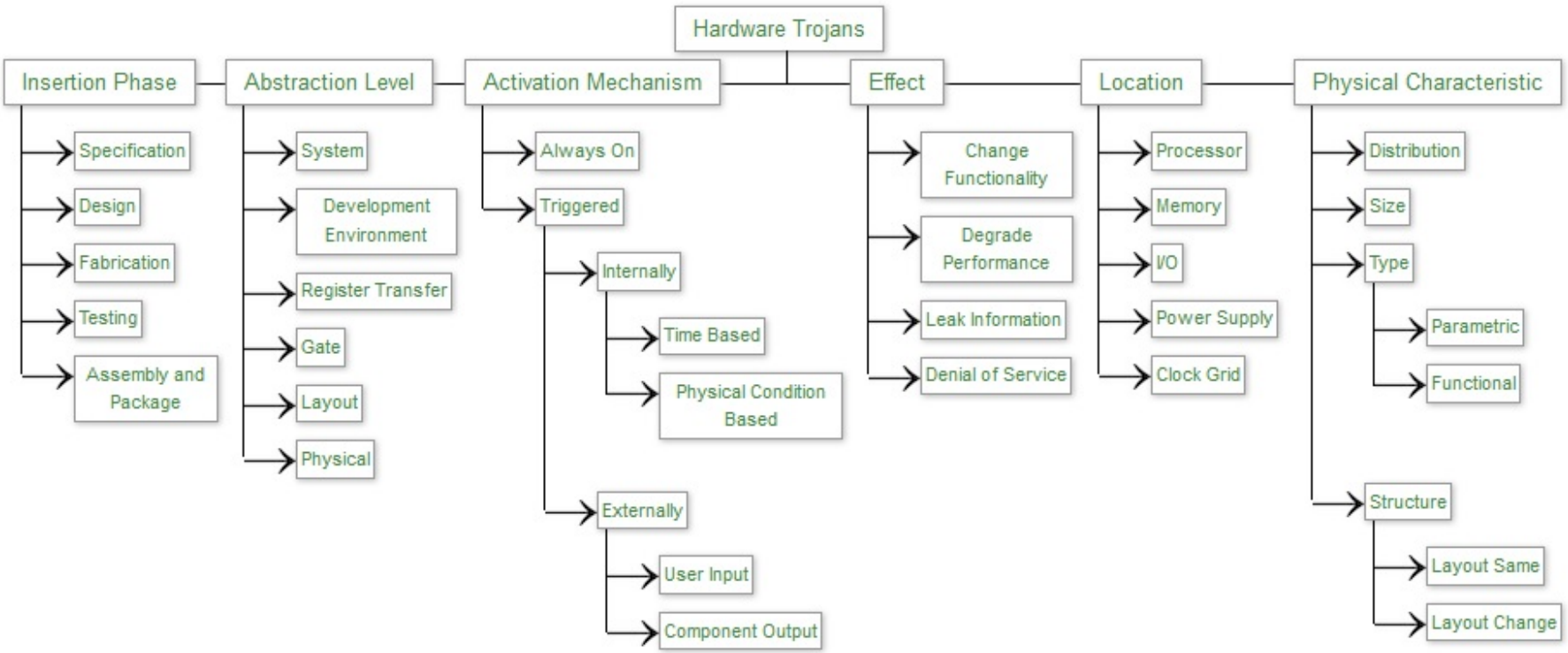
- Taxonomy : tree where each branch defines a different property
- In the ideal case, a specific HT must be on only one leaf of the tree

Benefits of the taxonomy

- Systematic study of their characteristics
- Specific detection methods for each HT class
- Benchmark circuits for each class

- Best existing taxonomy : Trust-Hub

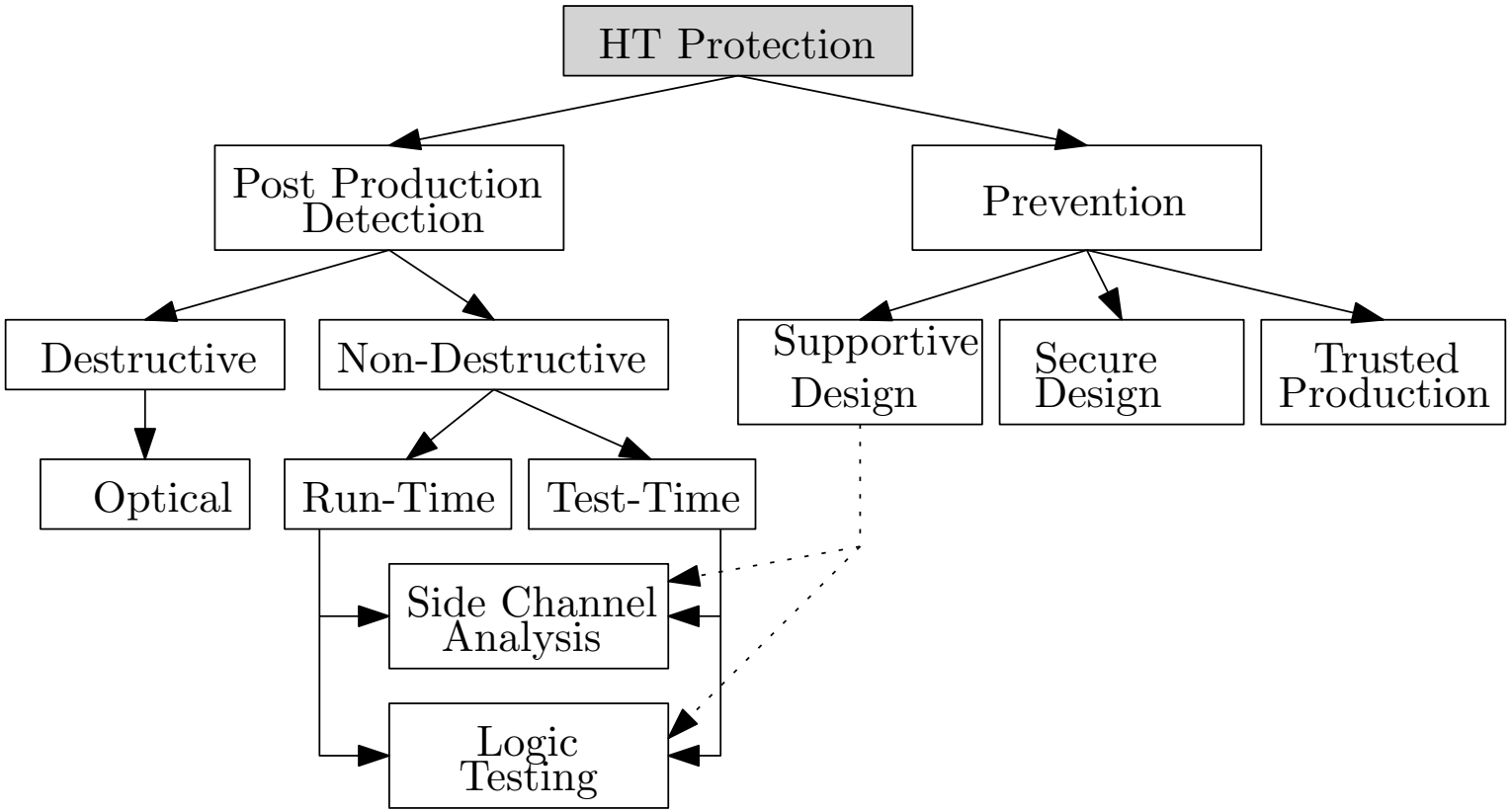
Trust-Hub Taxonomy



Factoring the Taxonomy

- 4 (effects) $\times$ 5 (locations) $\times$ 5 (insertion phases) $\times$ 6 (abstraction levels) $\times$ 5 (activation mechanisms) $=$ 3000 different HTs !
- Very rich taxonomy !
- Impossible to implement them all, and then detect them
- $\Rightarrow$ Factoring this taxonomy
- Total : $\sim$ 100 HTs

HT Detection Methods Overview



- No method is 100% successfull !

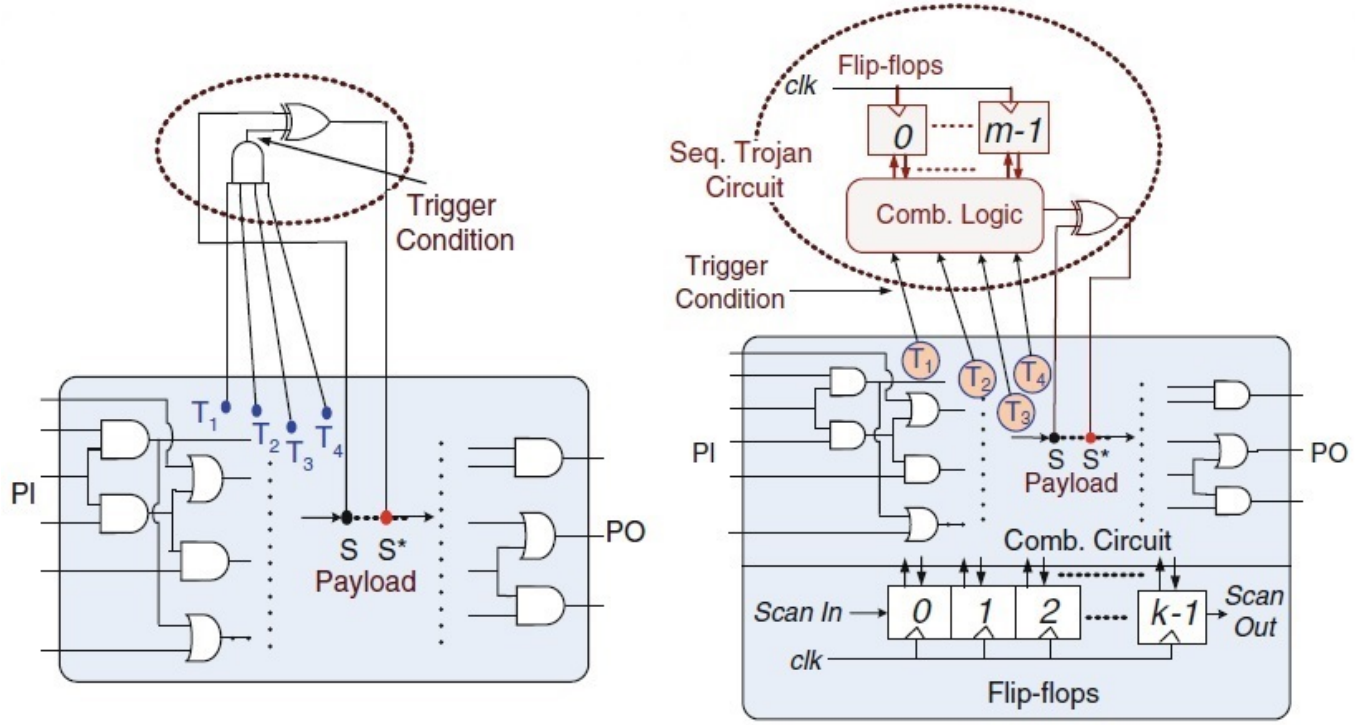
Detect HTs ? Not so easy...

1. Systems on Chip are more and more complex, and detecting a small malicious modification is difficult
2. Reverse-engineering inspection is costly and difficult
   - No guarantee that the remaining ICs are HT-free
3. By nature, HTs are designed to be stealthy
   - Not easily detectable with conventional logic testing
4. By nature, HTs are small to be not easily detected by optical analysis
   - Difficult to detect them with side-channel (power consumption, electromagnetic radiations, *etc.*) analysis

Test Generation (1/2)

- Conventional logic testing cannot be used to reliably detect HT
- Manufacturing defects (stuck-at-faults) $\neq$ HT effects
- Difficult to trigger a HT
  - *Time-bombs*
- Some HTs have no impact on functional outputs (*Trojan Side-Channels*)
- Vast spectrum of possible HTs

Test Generation (2/2)



- HTs are on low controllability and observability nodes for a rare triggering
- Extremely challenging to exhaustively generate test vectors for triggering a HT

Deterministic vs. Probabilistic Approach

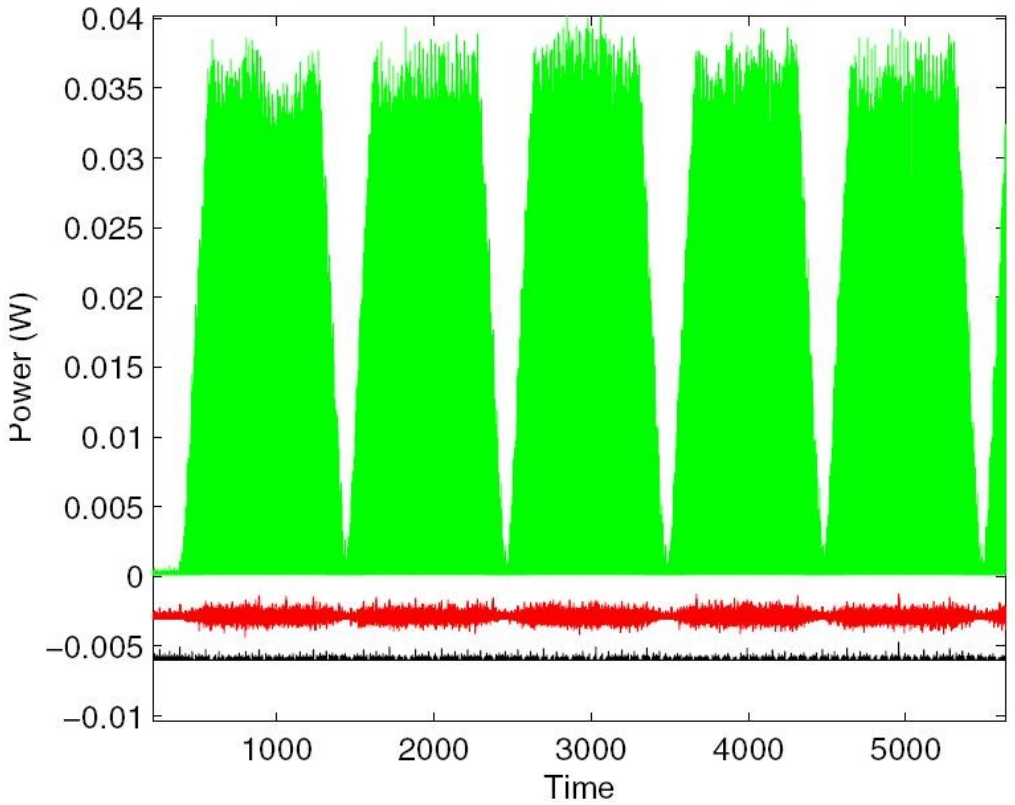
- Deterministic approach difficult
  - Many possible HTs
  - Function of some IC nodes
  - $\Rightarrow$ Exhaustive enumeration impossible
- Statistic approach :
  1. Find rare events in the circuit
  2. Get a list of HTs which can be inserted
  3. Generate test vectors and estimate their coverage
  4. $\Rightarrow$ Set of high quality test vectors
- 85% reduction in testset length compared to a random approach, but less efficient with big triggers and takes a long time

Side-Channel Analysis

- Any HT in the IC should modify its leakage current (IDDQ), dynamic power trace (IDDT), path-delay characteristic, ElectroMagnetic (EM) radiation.
- Don't need to trigger a HT for measuring its effects
- Test vectors generation easier than for logic testing
- Needs HT-free circuits
  - Get side-channel measurements and then *reverse-engineering* to check if the IC is HT-free
- If so, the measurements become a reference, and we can then compare the side-channels of the other circuits

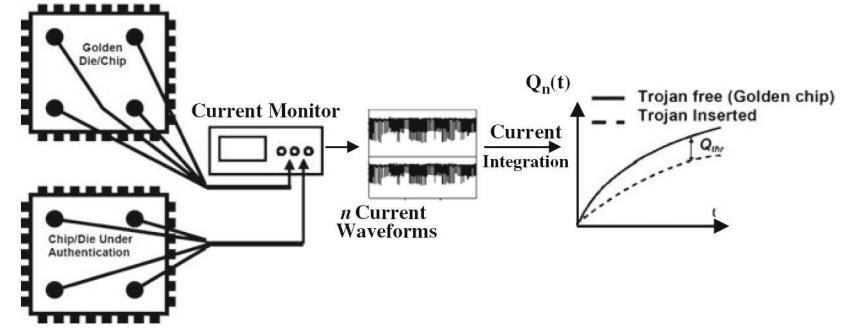
Global Side-Channel Analysis

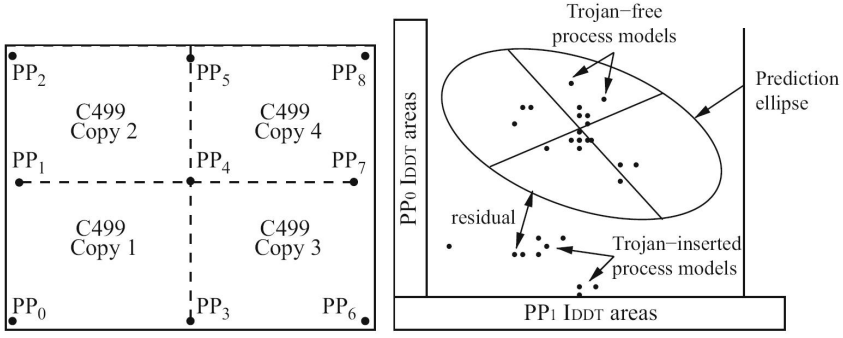
- Green : RSA signal
- Red : Process noise (offset)
- Black : HT signal (offset)



Local Side-Channel Analysis

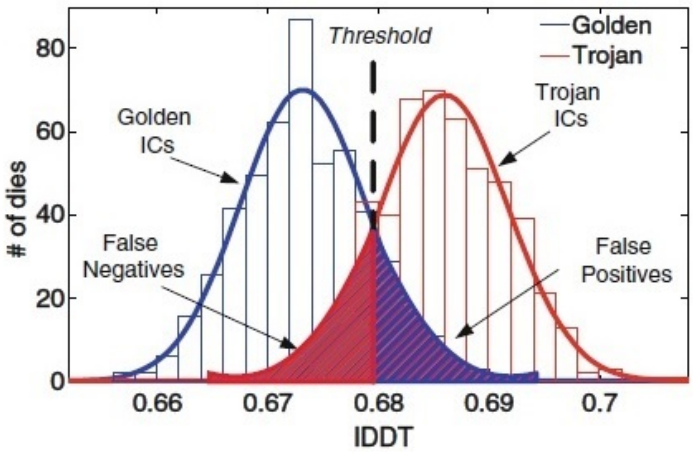
- Local Side-Channel Analysis more efficient than global ones
- Needs again HT-free circuits

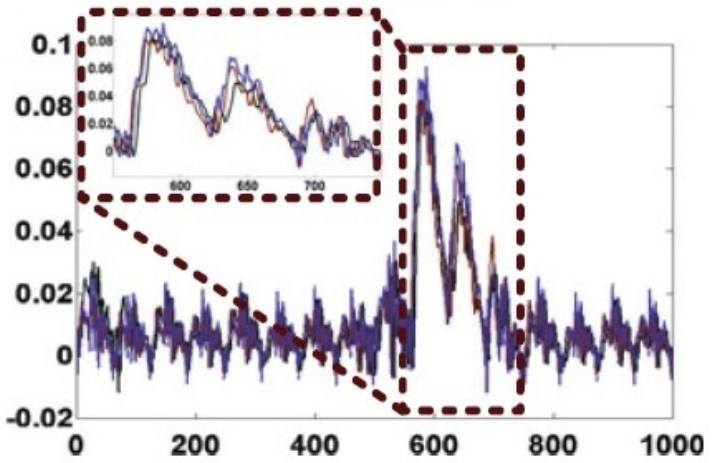


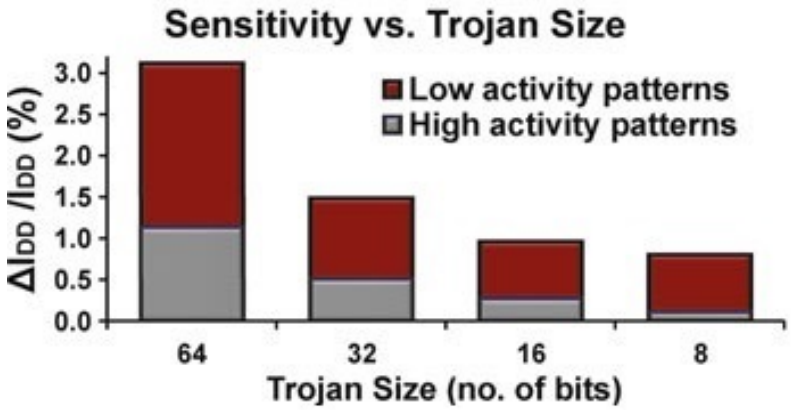


- Maximize/Minimize the activity of some IC areas

Noise and Sensitivity







Some Subtleties

- Added circuitry for the HT detection must not be infected itself
  - At best, the added circuitry is disabled (*e.g.*, fault countermeasure)
  - At worst, it can be turned into a *backdoor* (*e.g.*, scan chain)
- A HT triggering logic can exploit the "*Test/Scan Enable*" control line to disable itself
- Parametric HTs very difficult to detect

Summary

	Logic testing approach	Side-channel approach
Pros	(a) Effective for small Trojans	(a) Effective for large Trojans
	(b) Robust under process noise	(b) Test generation is easy
Cons	(a) Test generation is complex	(a) Vulnerable to process noise
	(b) Large Trojan detection challenging	(b) Small Trojan detection challenging

- Complementary methods
- Combine test-time and run-time methods
- Modify the IC for assistive and preventive methods
  - $\Rightarrow$ Design for Hardware Trust

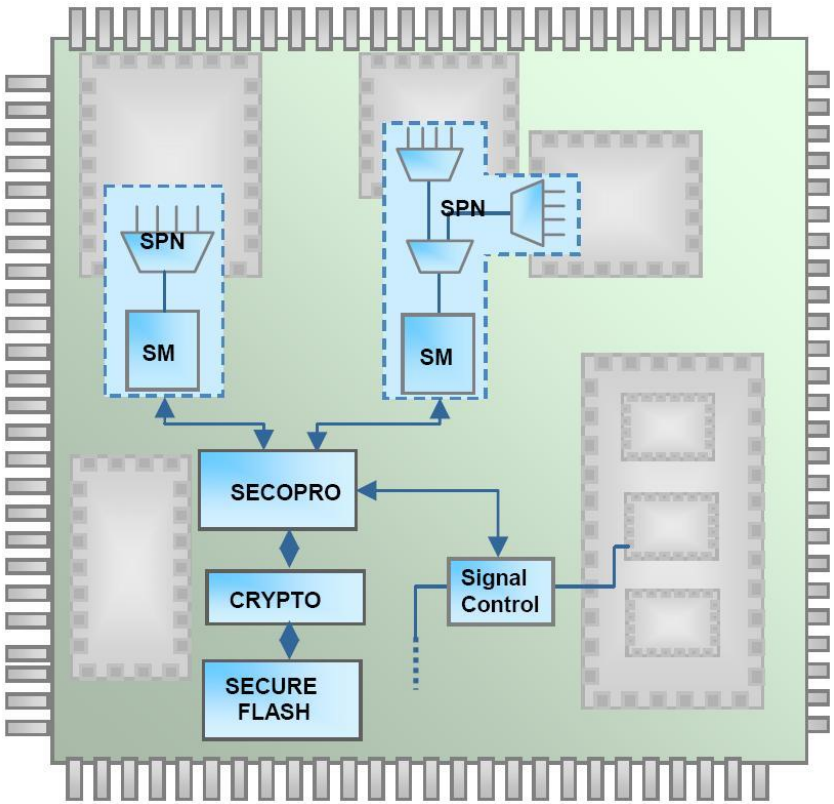
Introduction

- To improve HT detection rate, modify the IC
- $\Rightarrow$ Design for Hardware Trust
  - Prevent from the insertion of HT
  - Ease side-channel analysis and logic testing
- 4 main methods :
  - Delay-Based Methods
  - Rare Event Removal
  - Design for Trojan Test
  - Proof-Carrying Hardware
- Run-Time Detection Methods

Run-Time Methods

- Last line of defense
- On-line monitoring of the IC in real-time, for checks :
  - Critical operations,
  - Idle mode,
  - Security policies,
  - Performance or availability of some units,
  - *etc.*
- Costly

Run-Time Methods



- Disable one suspect block or force one operation
- SPN : *Signal Probe Network*
- SM : *Security Monitor* ($\sim$ FSM)
- SECOPRO : *Security and Control Processor*
- Configurations ciphered and stored in secured Flash memory
- Overhead ?

A French Project : HOMERE

- FUI14 (2012-2015) : HOMERE project
- Large companies
  - Cassidian CyberSecurity, Gemalto
- Small company
  - Secure-IC
- Academic partners
  - ARMINES, CEA-LETI, LIRMM, Télécom ParisTech
- Governmental help and support
  - ANSSI, (DGA)

HOMERE : First Results

- Infection of Benchmark Circuits
- General Side-Channel Test-Bench
- HT Detection via Visual Inspection
- Internal Delays Extraction by Fault Analysis

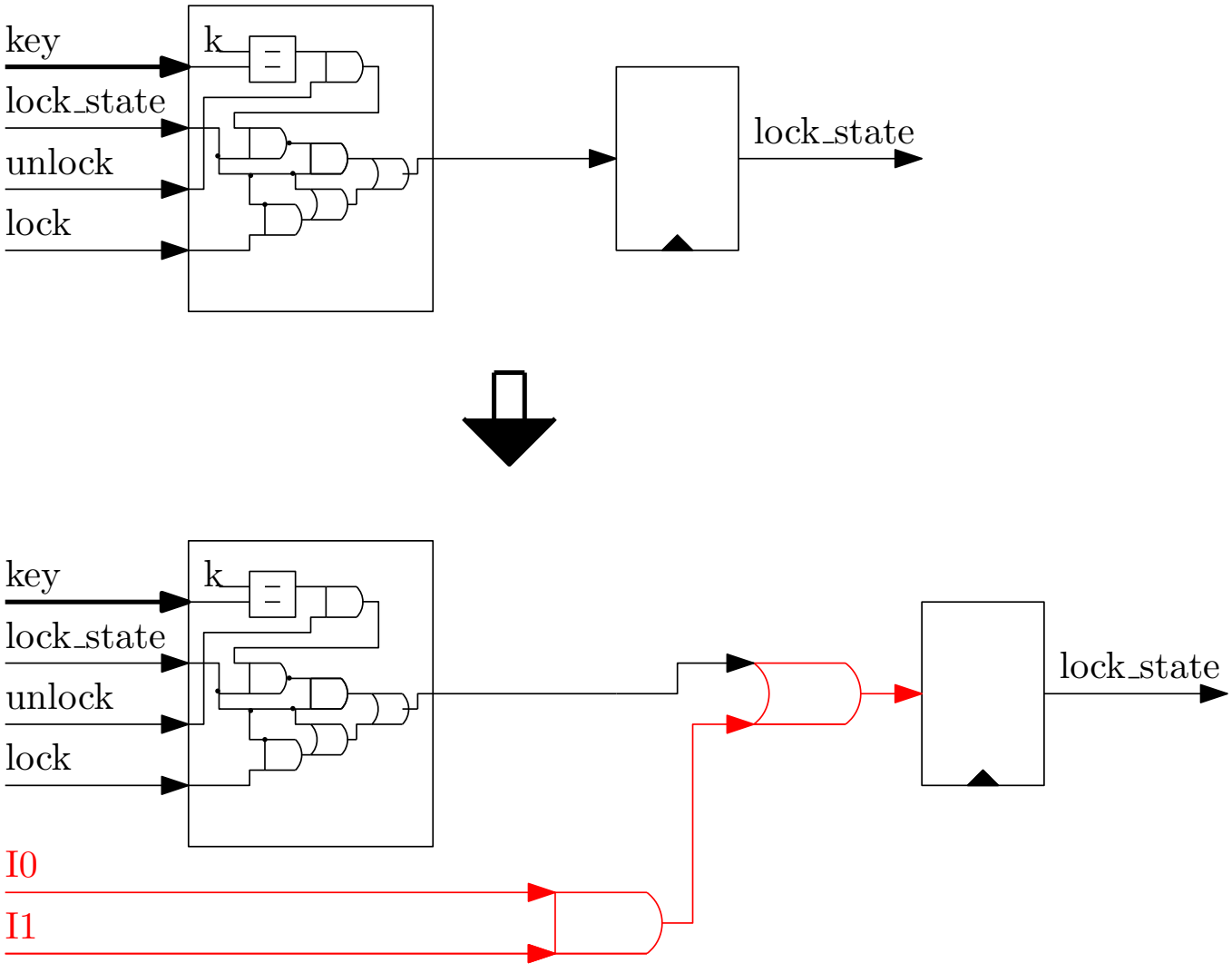
HOMERE : First Results

- Infection of Benchmark Circuits
- General Side-Channel Test-Bench
- HT Detection via Visual Inspection
- Internal Delays Extraction by Fault Analysis

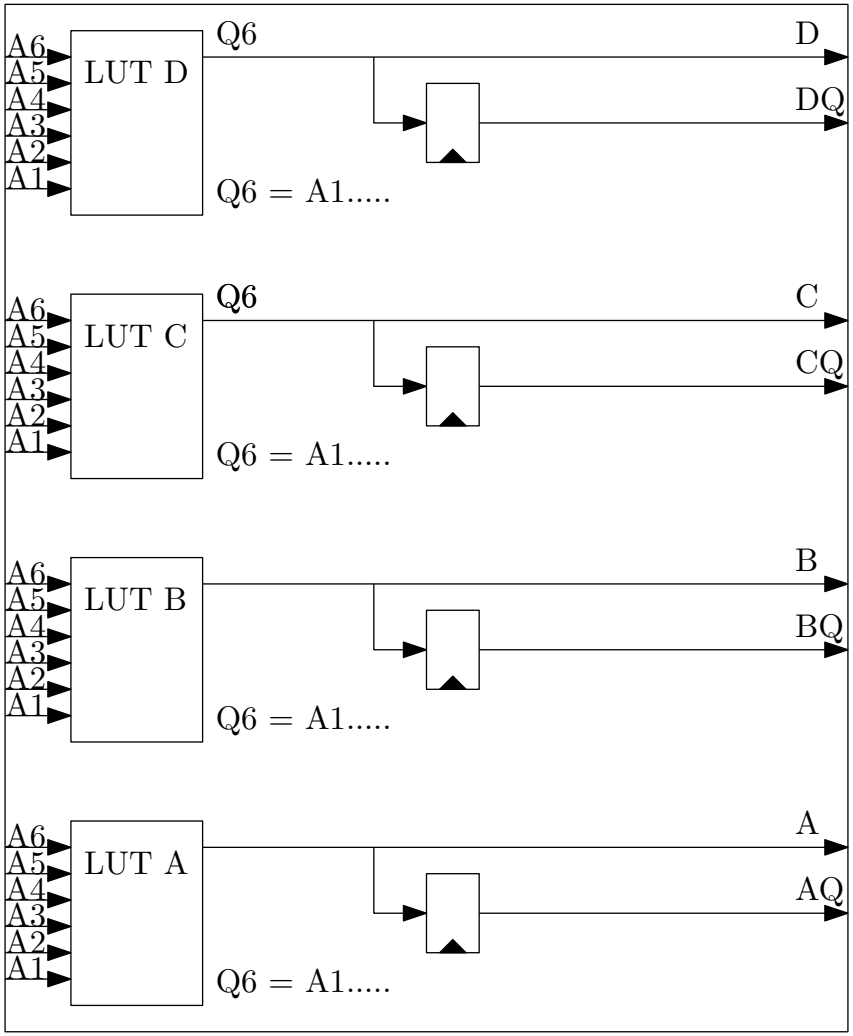
Infection of Benchmark Circuits

- Can be done at RTL level (VHDL)
  - But it will greatly change the final layout
  - Trust-Hub website suggest to implement the HT in VHDL level, place and route the circuit, and then delete the HT
  - Quicker than adding HT manually, but we will get a pair of (HT-free/Infected circuit) for each HT
- Will be done at netlist level
  - We have a common reference for each HT
- Manipulation of .ngc files for Xilinx
- We can then modify :
  - LUT content
  - Routing
  - Configurations (FFs or LATCH, IBUF delays in IOB, *etc.*)

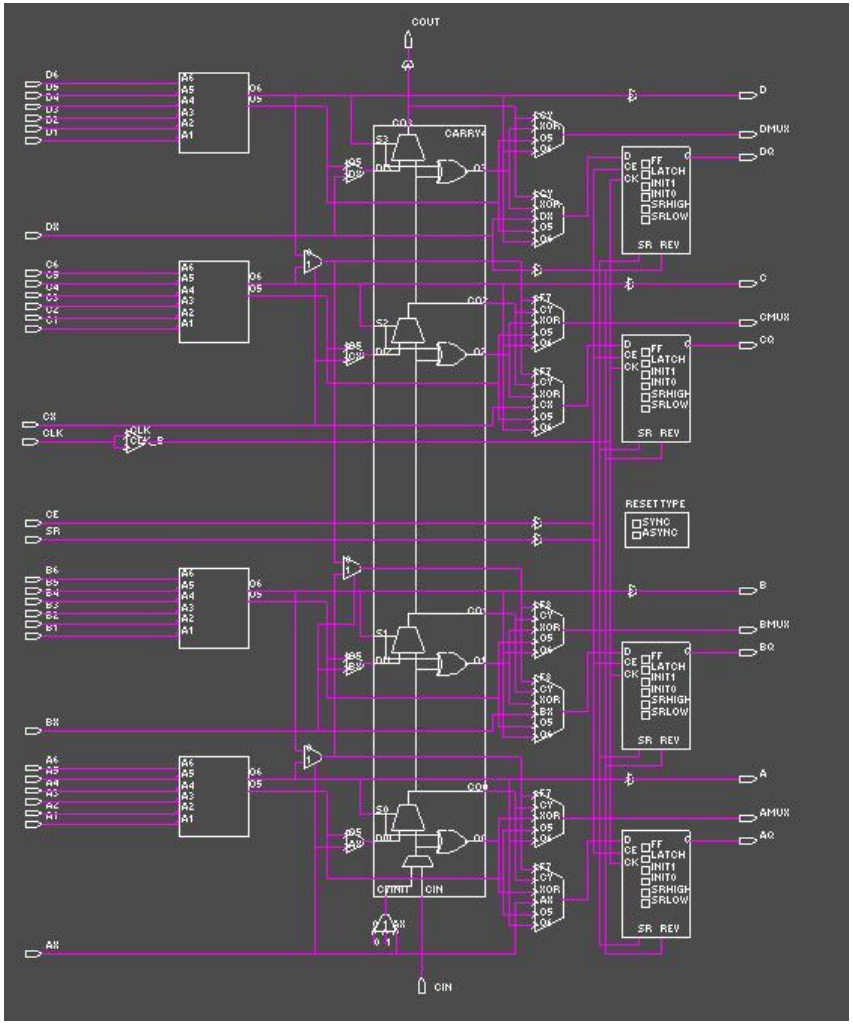
HT Inserted in RTL Mode



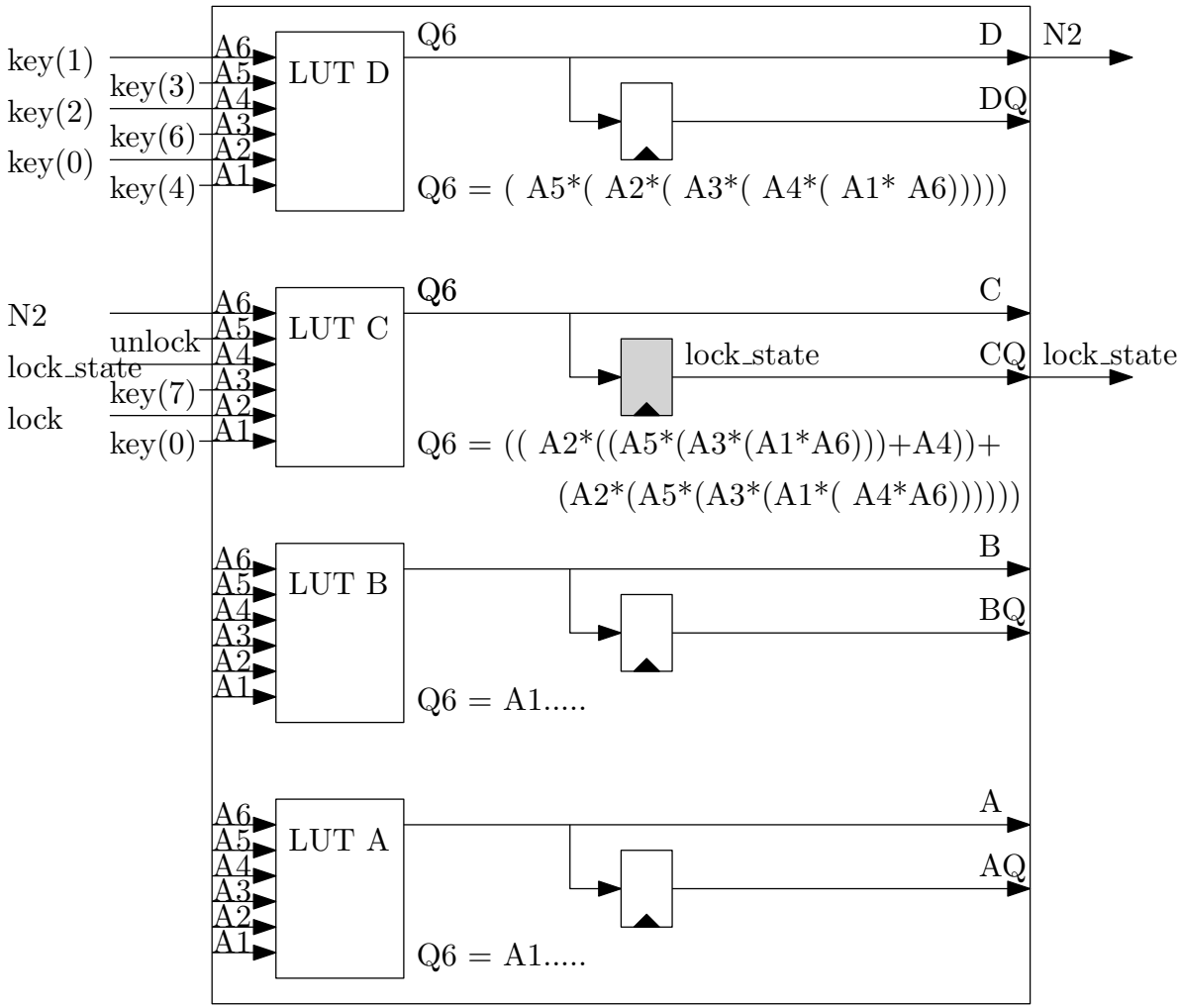
Slice of a Virtex-5 50T



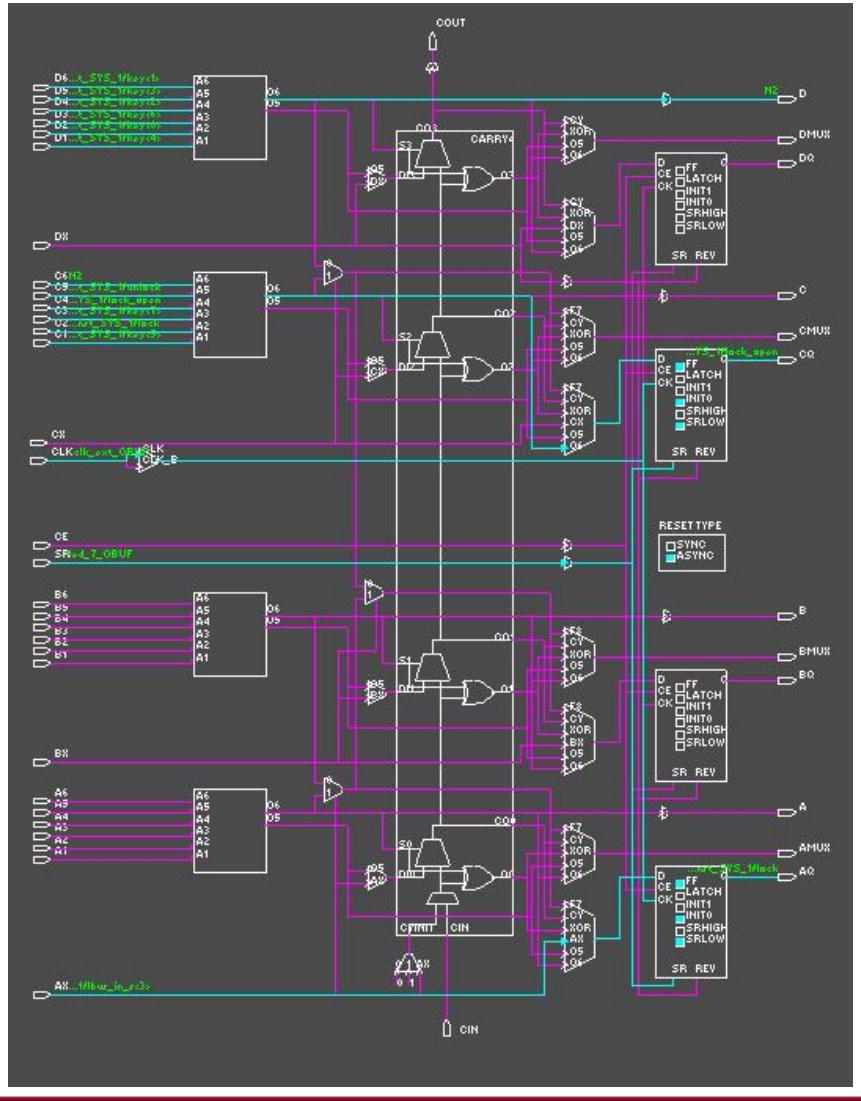
The same on “FPGA Editor”



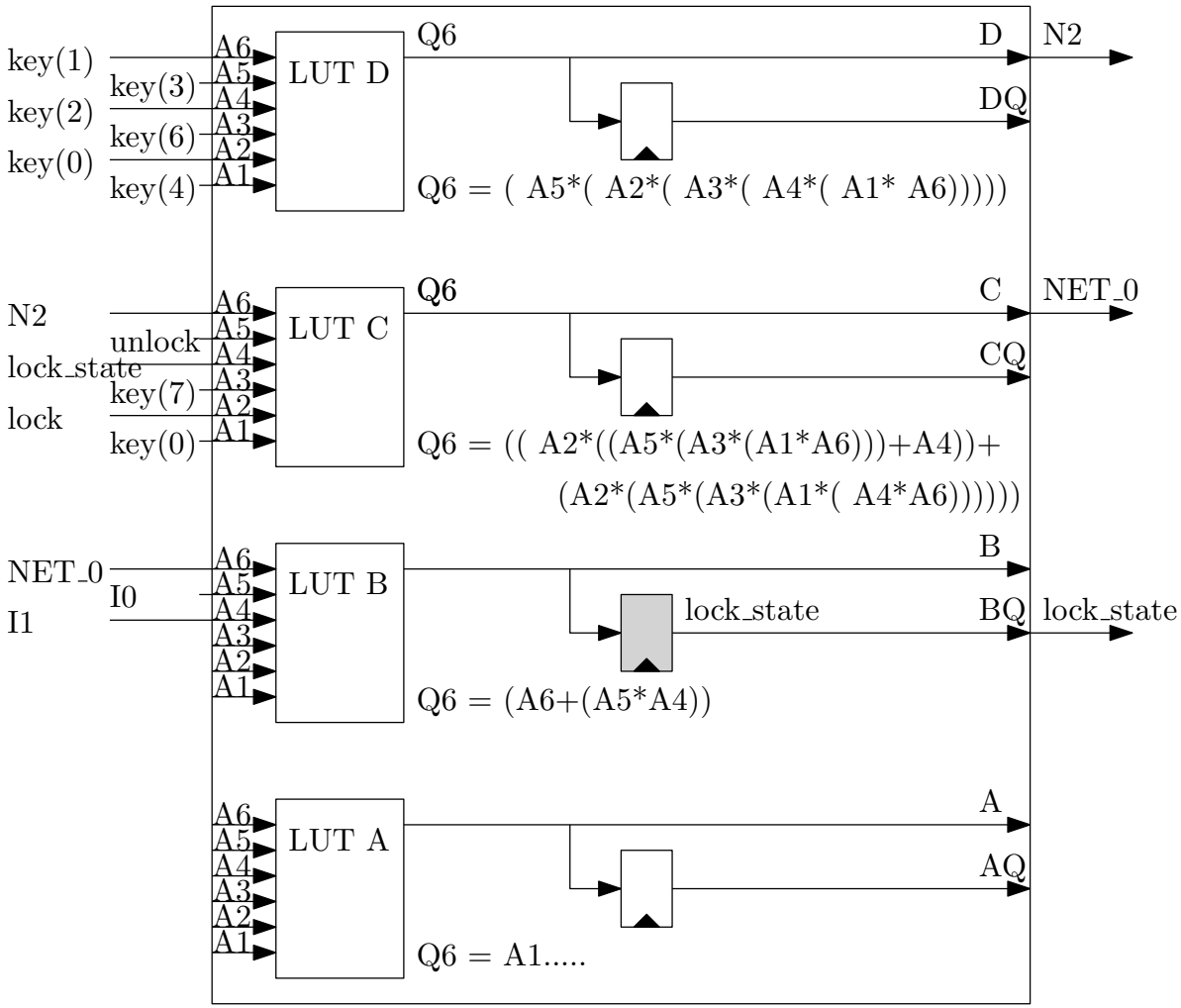
Non-Infected Circuit



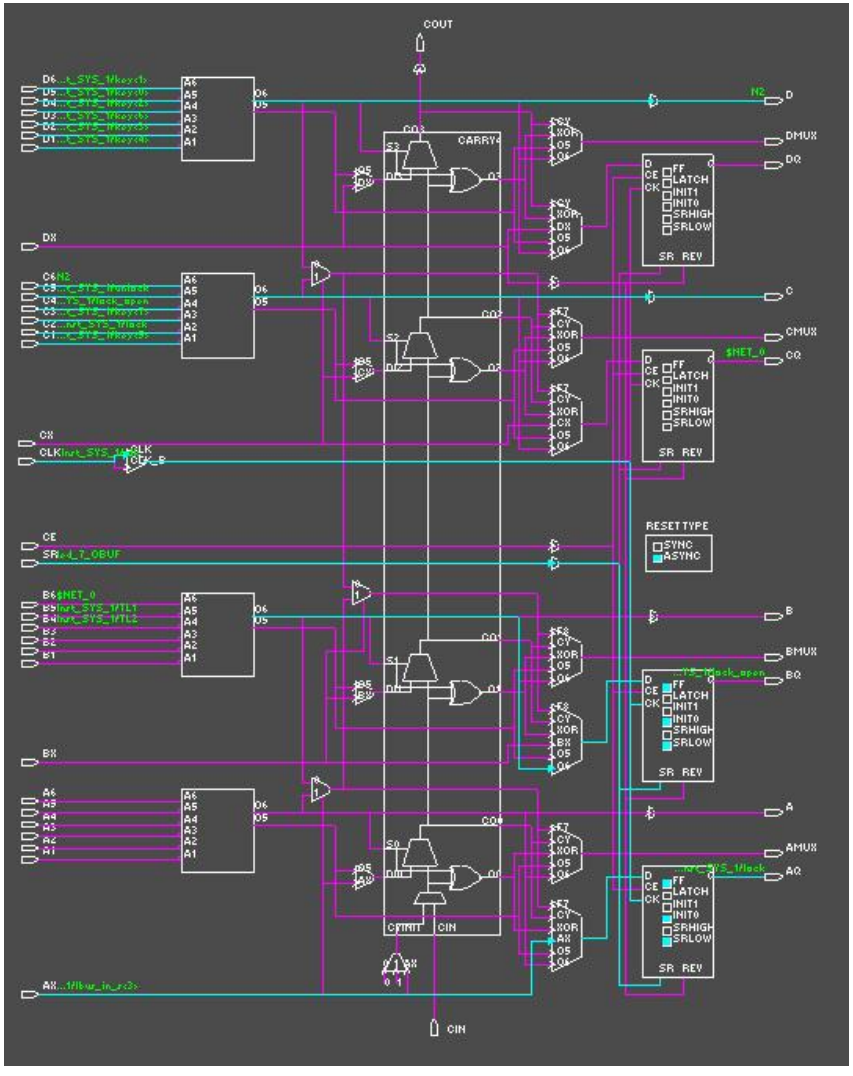
Non-Infected Circuit



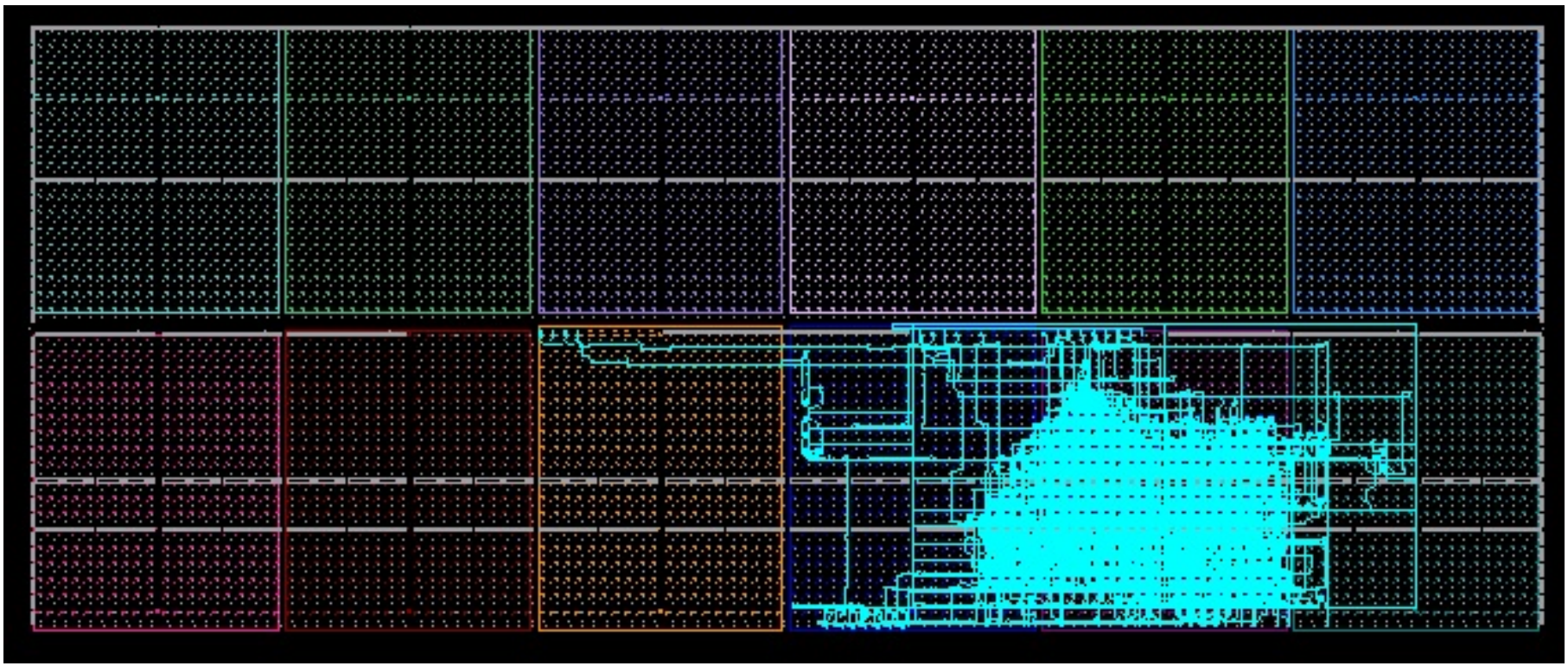
Infected Circuit



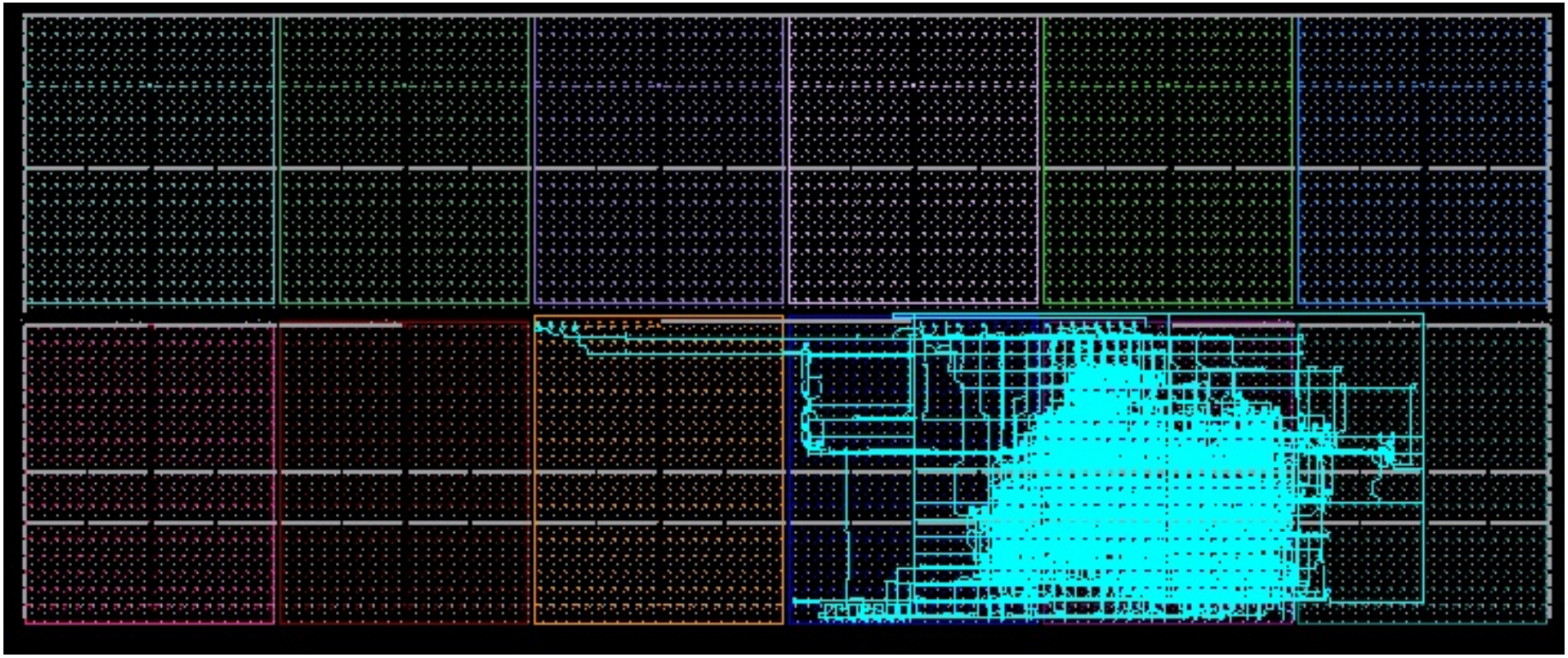
Infected Circuit



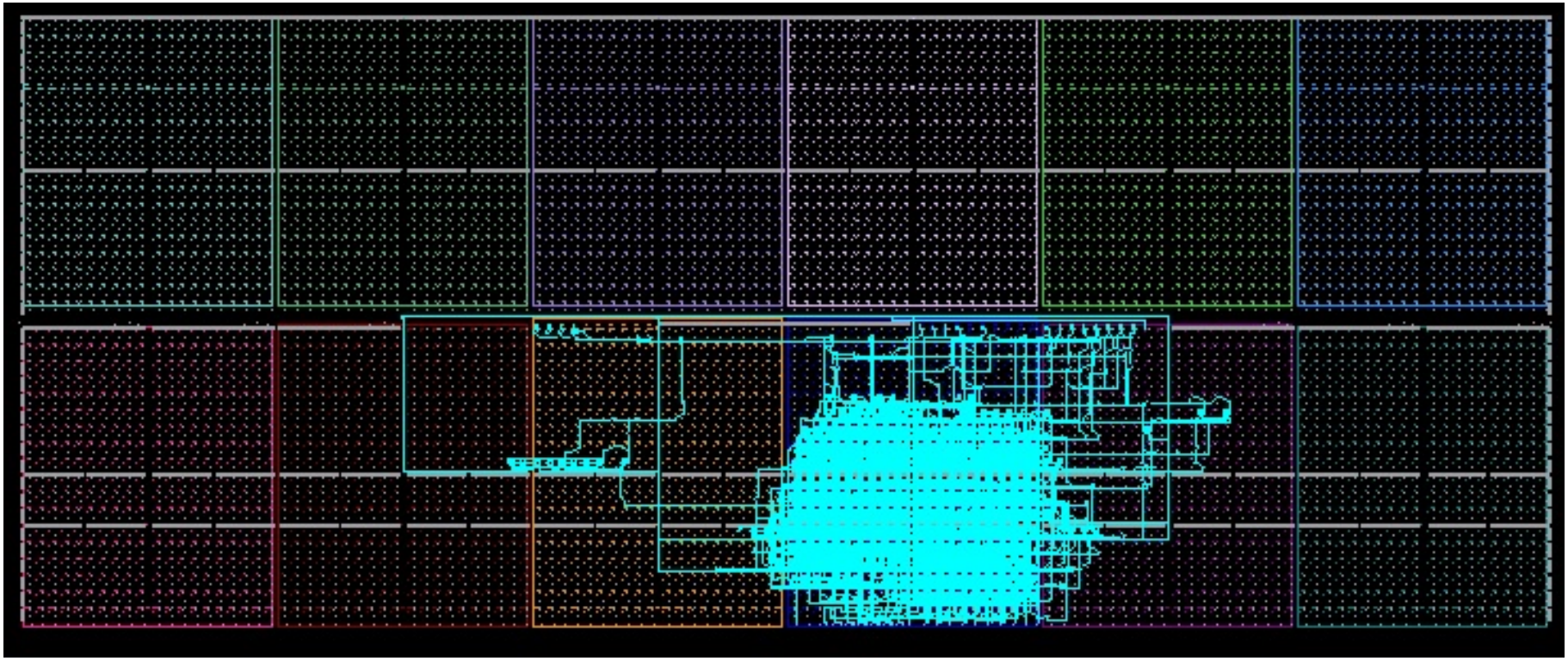
Original Circuit



Infected Circuit at Netlist Level



Infected Circuit at RTL Level



- Trojan Trigger : 32-bit counter in the I/O of an AES. Incremented at each clock cycle. HT activated when counter = FFFFFFFF. After activation, no more ciphertext will be sent to the output.

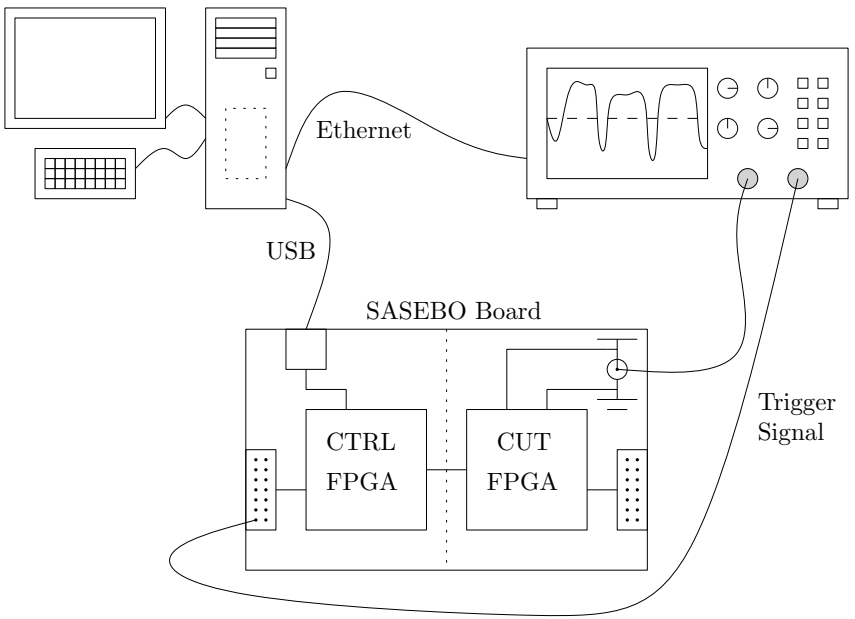
HOMERE : First Results

- Infection of Benchmark Circuits
- General Side-Channel Test-Bench
- HT Detection via Visual Inspection
- Internal Delays Extraction by Fault Analysis

Our Goals

- We want :
  - to understand the development process of an HT,
  - a list of candidates HTs,
  - to implement these HTs,
  - to check that inserted HTs can be triggered.
- We want a side-channel test-bench which :
  - is generic,
  - allows to test different circuits...
  - ...infected by different HTs.
- Side-chAnnel Standard Evaluation BOard (SASEBO)

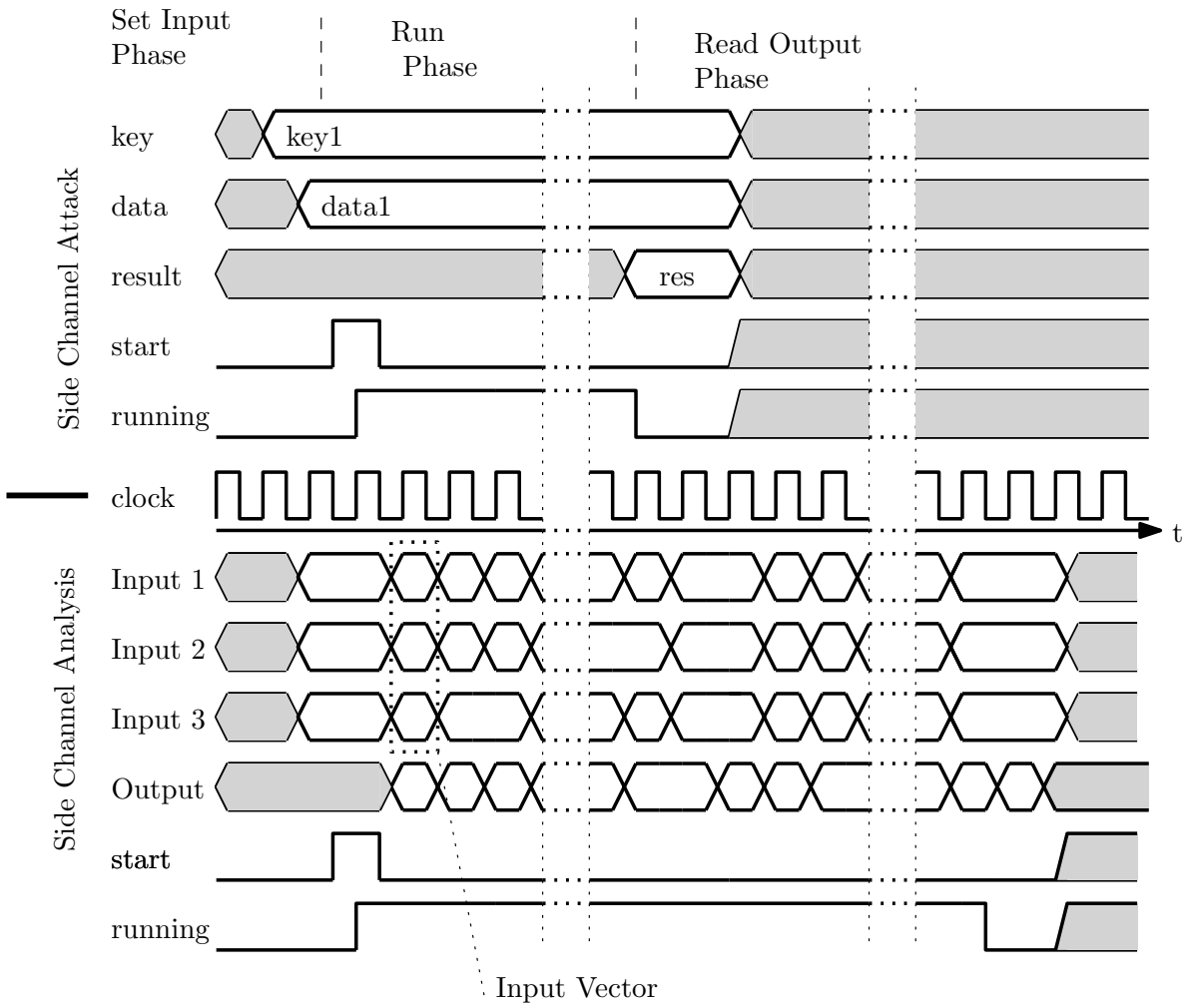
Overview of SASEBO



- 2 FPGAs :
  - 1 for the *Circuit Under Test* (CUT),
  - 1 for the control (can be used for different CUTs).
- USB connection between PC and SASEBO

Side-Channel Analysis

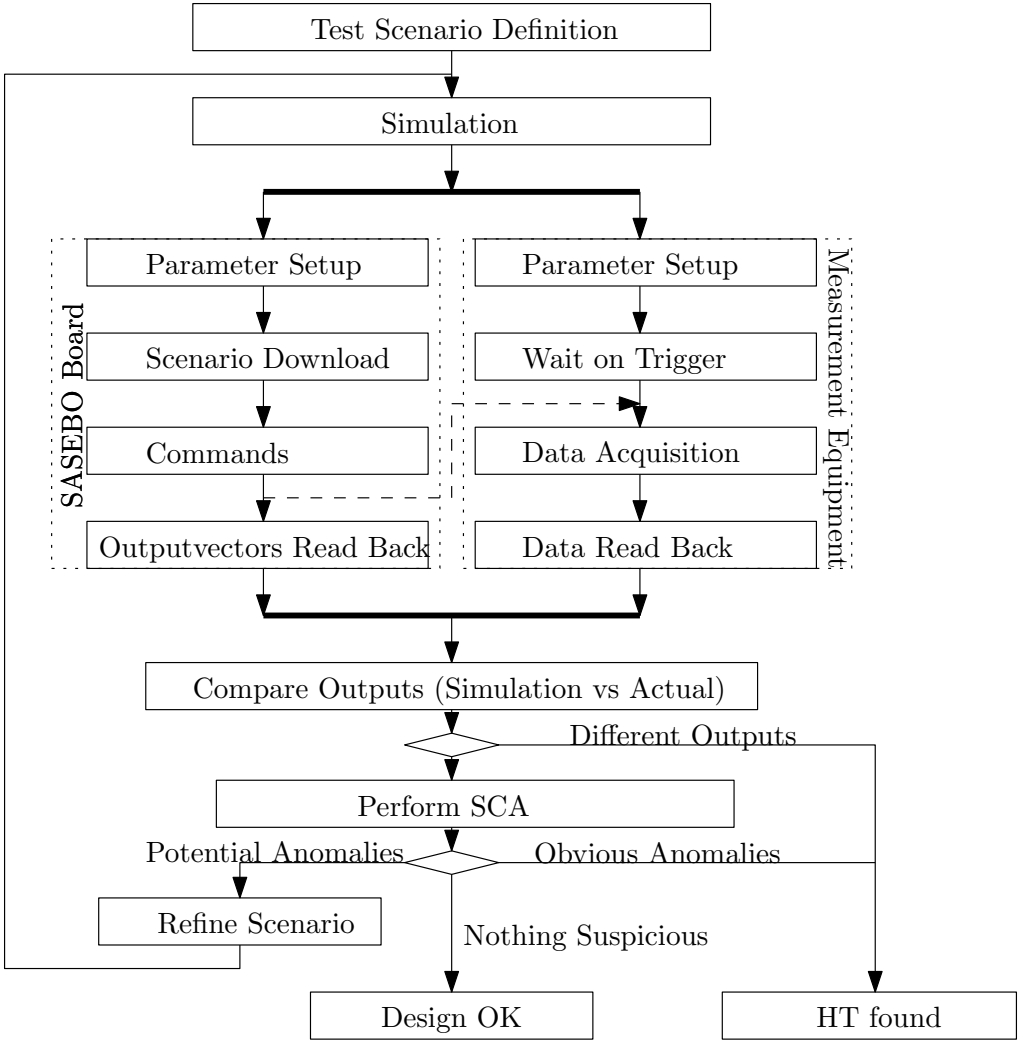
- Attack (for retrieving keys) $\neq$ Analysis (for detecting HTs)



Lessons for our Test-Bench

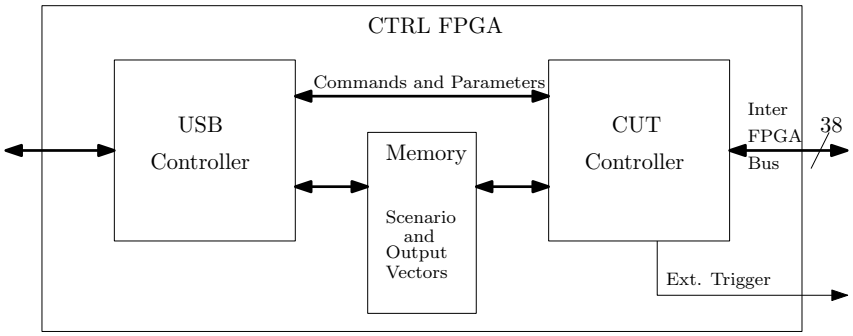
- In general :
  - we must generate complex sequences of input vectors,
  - we have to get intermediate outputs,
  - we want real-time I/O processing.
- To trigger an HT, and for detect it, we need :
  - to wait for a long time,
  - to react according to the behavior of the tested circuit.
- For side-channel analysis :
  - Dynamic triggering of the measurements.

Workflow


Test Scenario Definition
Simulation
SASEBO Board
Parameter Setup
Scenario Download
Commands
Outputvectors Read Back
Measurement Equipment
Parameter Setup
Wait on Trigger
Data Acquisition
Data Read Back
Compare Outputs (Simulation vs Actual)
Different Outputs
Perform SCA
Potential Anomalies
Obvious Anomalies
Refine Scenario
Nothing Suspicious
Design OK
HT found

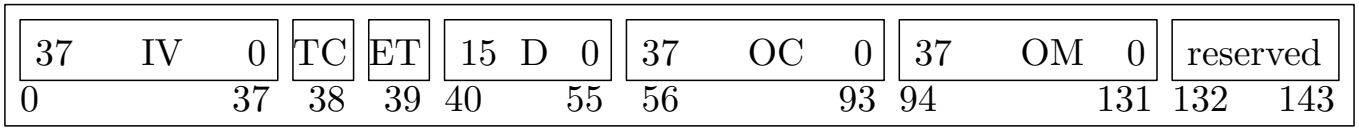
Test Scenario Description

- USB communication of the test scenario file
- Stored in the memory (BRAM) of control FPGA



- 3 options for sending the next input vector
  - Immediate
  - Time condition
  - Output condition
- External triggering flag
- Data format ?

Data Format



- IV : *Input Vector*,
- TC : *Transition Condition*, time or output conditions to send the next IV,
- ET : *External Trigger*, sent to the oscilloscope for starting measurements,
- D : *Delay* : number of clock cycles to delay the next IV,
- OM : *Output Mask* : which bits we are looking at ?
- OC : *Output Condition* : values of these bits to send the next IV.

Test Scenario Format

Scenario Description

Parameters

Parameter 1	Value 1
Parameter 2	Value 2
...	... ...

Attributes

Name	Scenario Name
Version	Definition Version

Input Profile

1	IV	TC	ET	D	OC	OM	reserved
2	IV	TC	ET	D	OC	OM	reserved
3	IV	TC	ET	D	OC	OM	reserved
...	...	...	...	...	...	...	
n	IV	TC	ET	D	OC	OM	reserved

Commands

Command 1
Command 2
... ...

- Parameters : *tristate mask*
- Controller supporting this format validated

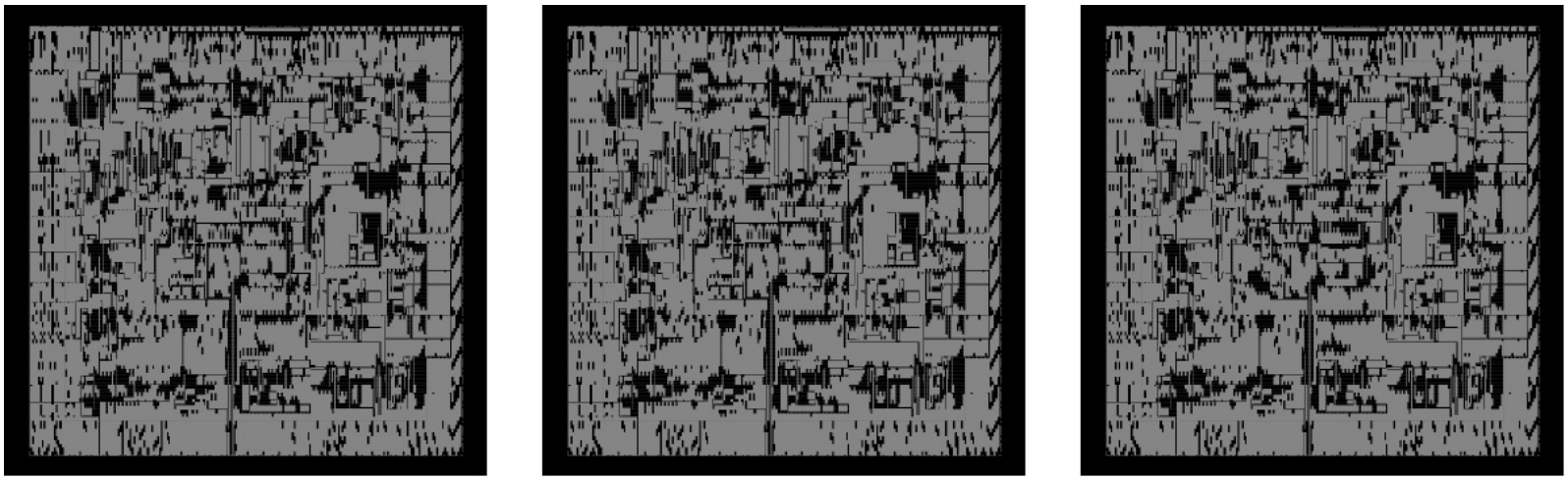
HOMERE : First Results

- Infection of Benchmark Circuits
- General Side-Channel Test-Bench
- HT Detection via Visual Inspection
- Internal Delays Extraction by Fault Analysis

Principle

- [Bhasin *et al.*, FDTC 2013]
- Study the effect of HT insertion at the layout level (GDSII)
- Is it possible to detect HTs *via* visual inspection ?
- CUT : AES-128
- HT : key leakage with fault injection (Piret/Quisquater attack) triggered on a specific plaintext
- Placement density of the circuit : 50% $\rightarrow$ 99%
- HT trigger size : 1 $\rightarrow$ 128 AND gates
- Cadence SOC Encounter

AES Layout (Metal6)



- AES Layouts for the 6th Metal Layer (1200 μm $\times$ 1200 μm) with placement density = 50% : (left) HT-Free AES, (middle) AES with 1 AND gate HT, (right) AES with 128 AND gates HT
- (Credits : Télécom ParisTech)

Results

- Preventive method : it is impossible to insert a HT in ECO mode if placement density $> 90\%$
- (*Post-mortem*) detection : visual correlation decreases when HT size and placement density of the circuit increase
- "Low Cost" way to detect HTs by the correlation between GDSII and circuit images
- More difficult to detect the very small HTs

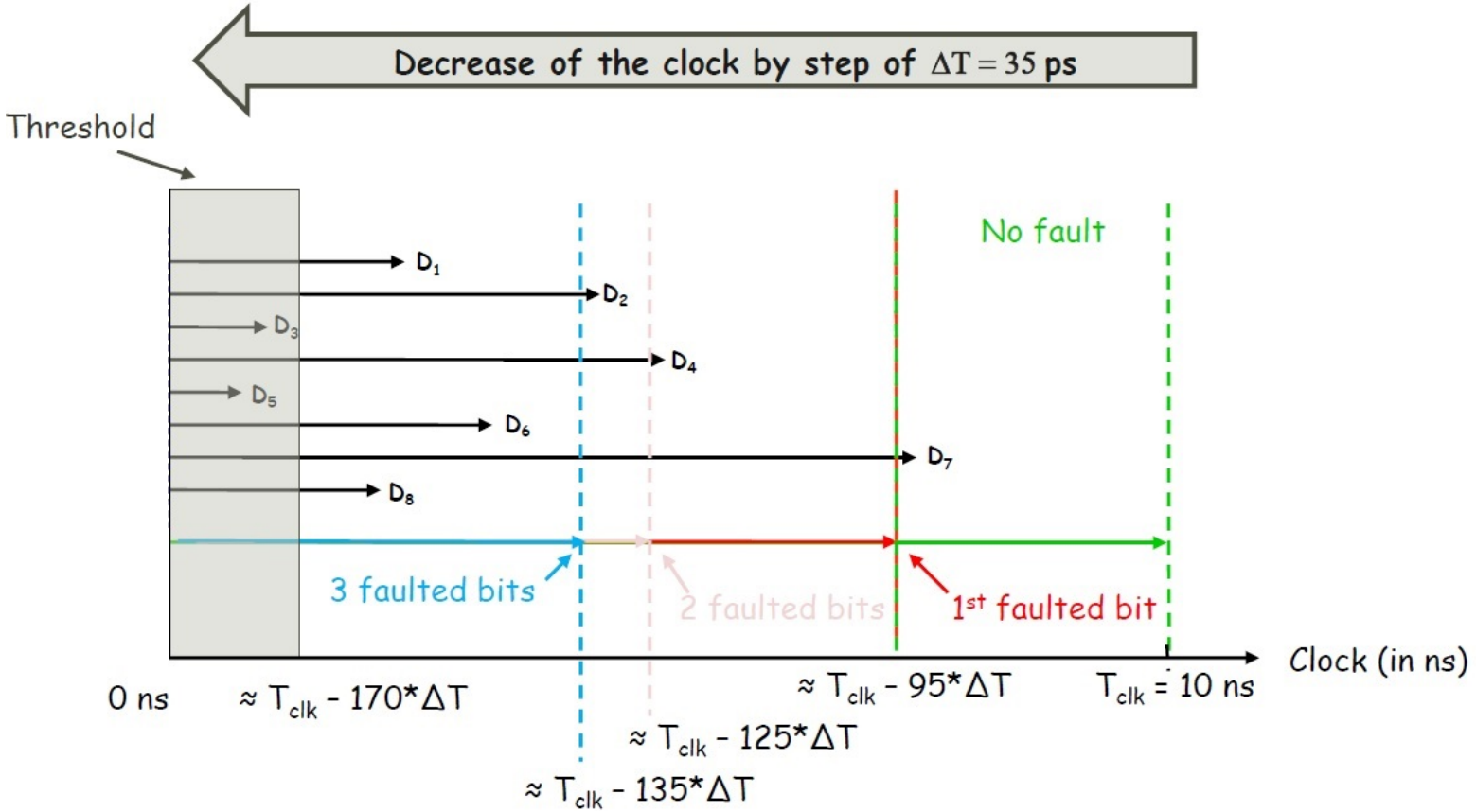
HOMERE : First Results

- Infection of Benchmark Circuits
- General Side-Channel Test-Bench
- HT Detection via Visual Inspection
- Internal Delays Extraction by Fault Analysis

Principle

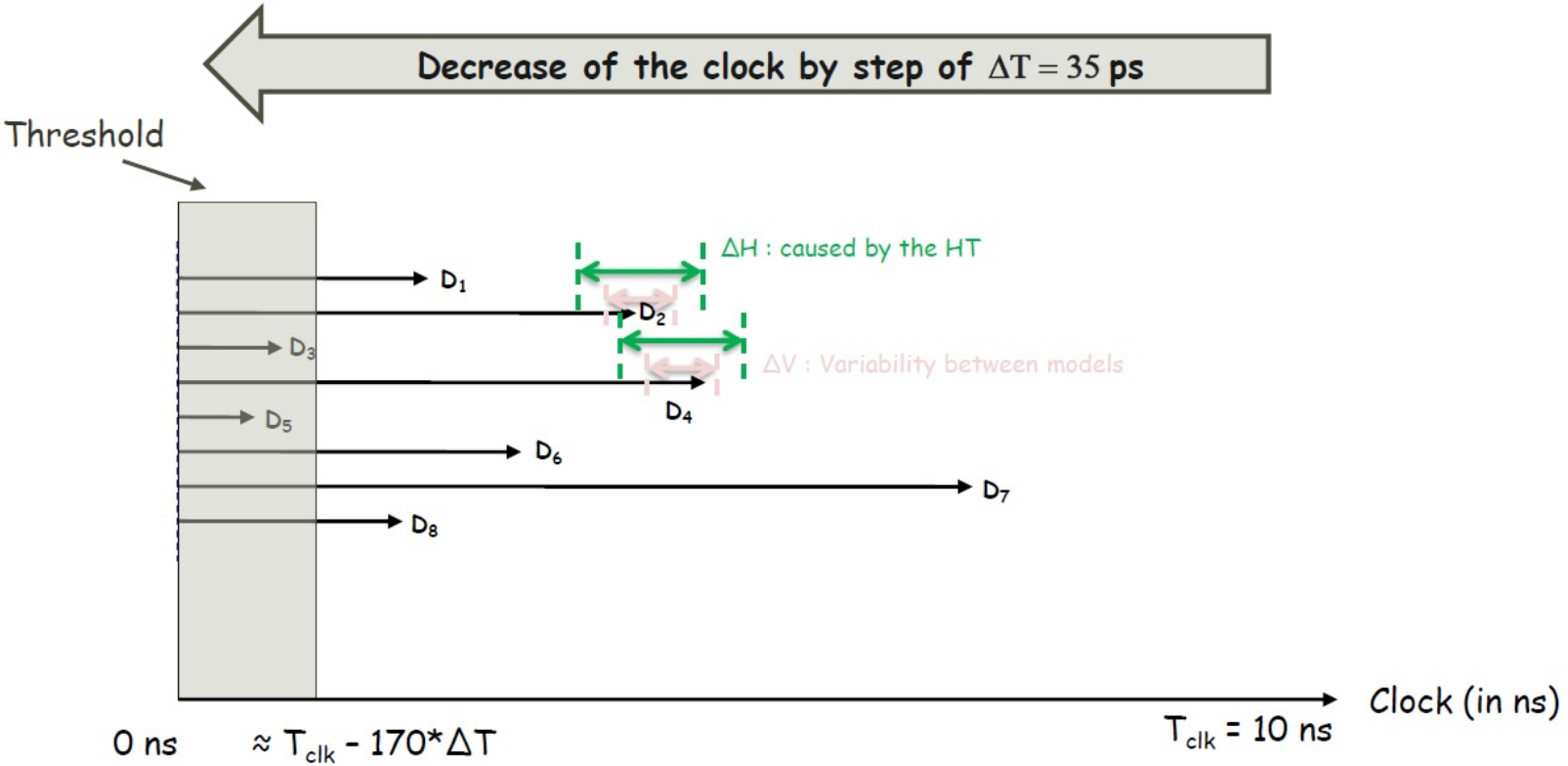
- [Exurville *et al.*, ReCoSoC 2013]
- An inserted HT will modify internal delays
- Idea : compare the fault sensitivity analysis of a genuine circuit and an infected one
- Glitches on external clock
  - The clock glitch is a local change of a period
  - The choice of the injection cycle is possible
- $\Rightarrow$ Setup time violations
- $\Rightarrow$ Metastability (non-deterministic behavior)

Fault Sensitivity Analysis in HT-Free IC



- (Credits : CEA-LETI)

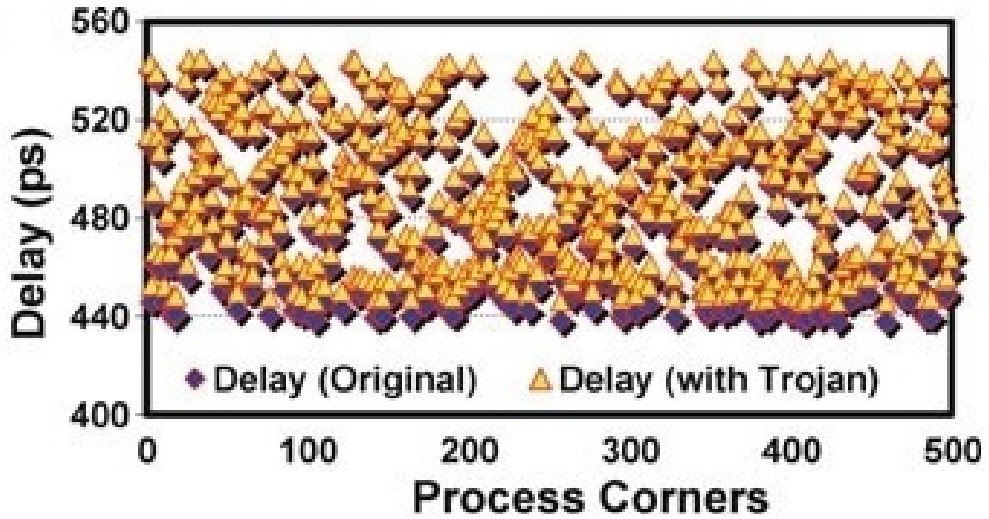
Fault Sensitivity Analysis in Infected IC



- (Credits : CEA-LETI)

First Results

- AES characterization thanks to the fault sensitivity analysis of each AES bit
- An inserted HT can influence the critical paths
- Challenges :
  - Process variations
  - HTs inserted in non-critical paths
  - Fault countermeasures

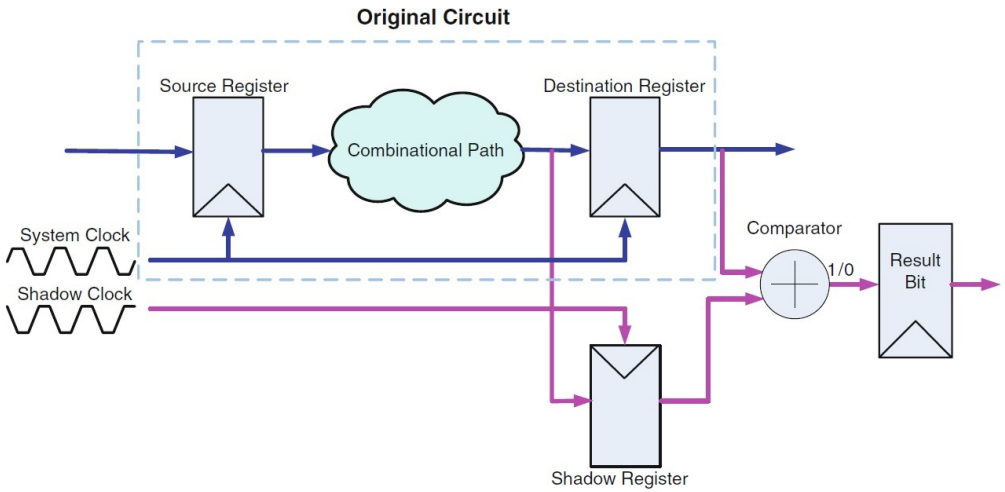


Conclusion

- Hardware Trojans are real threats for integrated circuits
- HT taxonomy is very rich
- No HT detection method of the state-of-the-art is 100% successful
- 3 lines of defense :
  - Design for Hardware Trust
  - Test-Time Methods
  - Run-Time Methods
- A French initiative : HOMERE project
- Very encouraging first results :
  - Infected benchmark circuits will be available soon
  - A common platform for side-channel analysis
  - A "low-cost" way to detect some HTs by visual inspection
  - A "low-cost" way to extract internal delays of ICs by clock glitching
- Other on-going works :
  - Logic test
  - Run-time HT detection

Thanks ! Questions ?

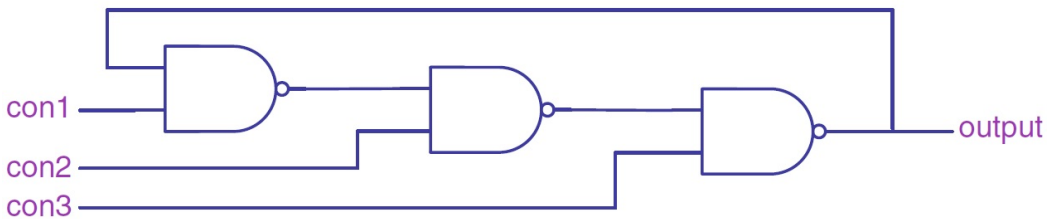
Shadow Registers



- Measurement of delays between registers
- Shadow Clock has a negative skew with respect to System Clock for characterizing the path delay
- Millions of paths ⇒ Big overhead

Ring Oscillators (1/2)

- Alternative to *shadow registers*
- Build new paths and measure the delays of these paths



- Small area
- Easy insertion
- Under normal operation, all the inserted ring oscillators will be muted to avoid power consumption

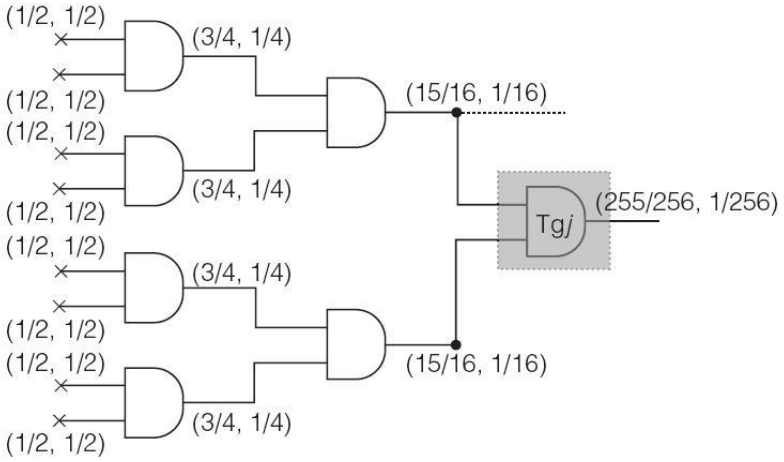
Ring Oscillators (2/2)

- Any malicious modifications to the original design woud also change parameters of pre-inserted ROs
  - Frequency change for the ROs
- How many ROs are needed where they should be located inside the chip ?
- Construct ROs from gates of the original design by inserting multiplexors, NAND gates and inverters
- "*On-chip*" frequency measurement modules

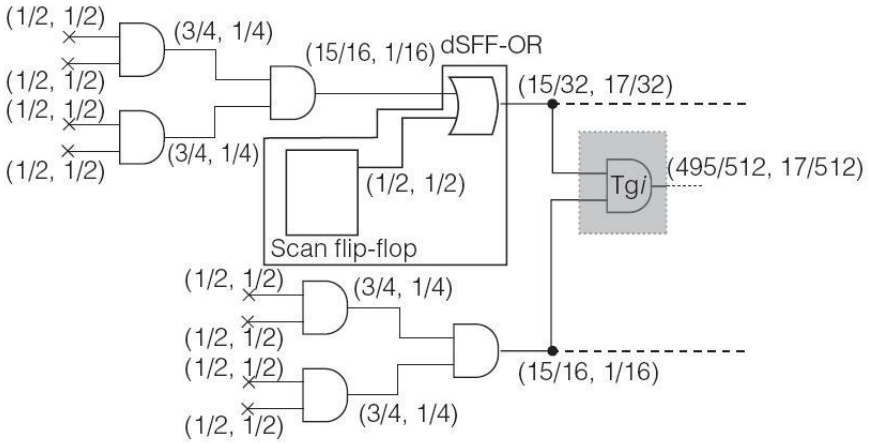
Drawbacks

- Difficult automation of RO insertion
- Easy to evade

Rare Event Removal (1/2)



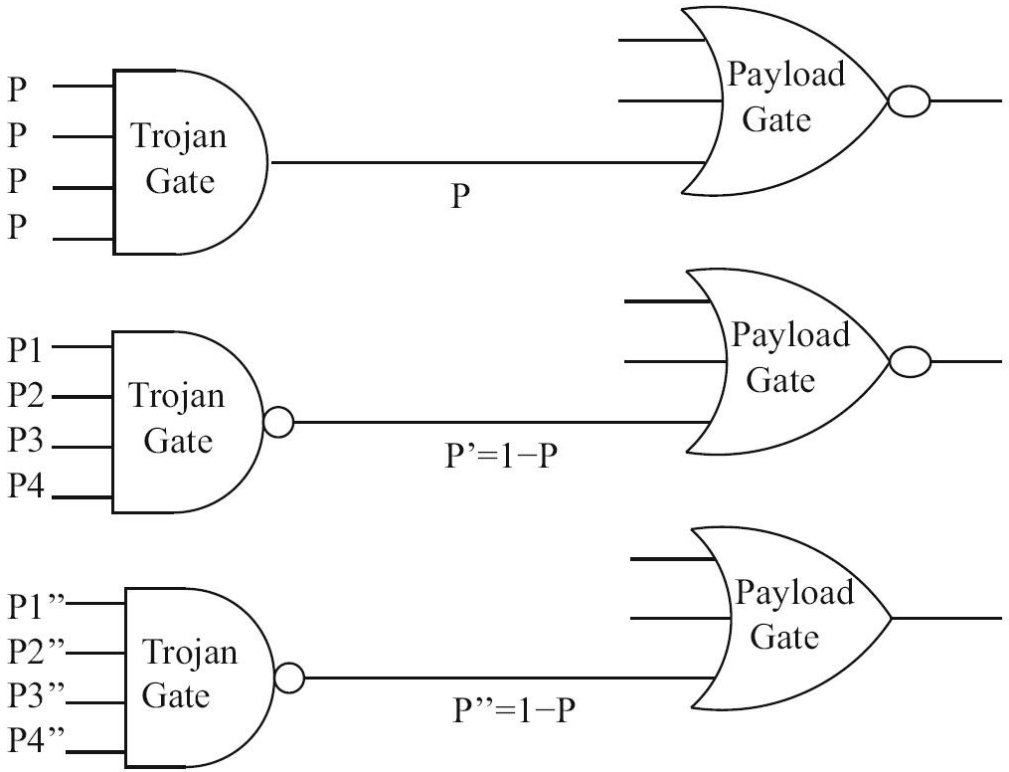
Transition probability at Trojan output = 255/65536
Average clock cycles per transition by GD = 255.6
Average clock cycles per transition by simulation = 250



Transition probability at Trojan output = 8415/262177
Average clock cycles per transition by GD = 30
Average clock cycles per transition by simulation = 33.4

×: Represents scan flip-flop or primary input

Rare Event Removal (2/2)



- Not adapted

Partenaires HOMERE

CASSIDIAN CYBERSECURITY

ARMINES

cea

gemalto
security to be free

Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier
LIRMM

SECURE IC
THE TRUSTED COMPUTING COMPANY

TELECOM ParisTech

oseo