SR2I301

Overview of Protections against IC Counterfeiting and Hardware Trojan Horses

Jean-Luc Danger

Outline

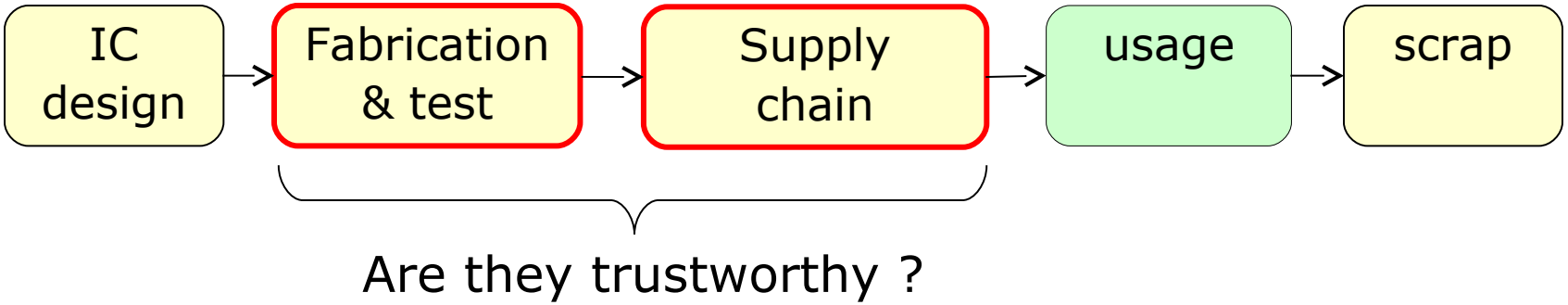
- IC Counterfeiting
  - Overview of the threat
  - Detection methods
  - Prevention methods
- Hardware Trojan Horses
  - Types
  - Detection methods
  - Prevention methods
- Conclusions

IC Counterfeiting is a reality

- The IC **Supply chain** (distributors, brokers,…) is an open door to counterfeit components [1] (SIA report)
- All sectors are impacted, including military[2] 5DoD report)
- Economic Harm
  - Reduction of the Original Component Manufacturer (OCM): market share: $169 billion in 2012 [3]
- Damage due to lack of reliability
  - The counterfeit circuit may be defective
  - Negative image of the OCM

IC counterfeiting

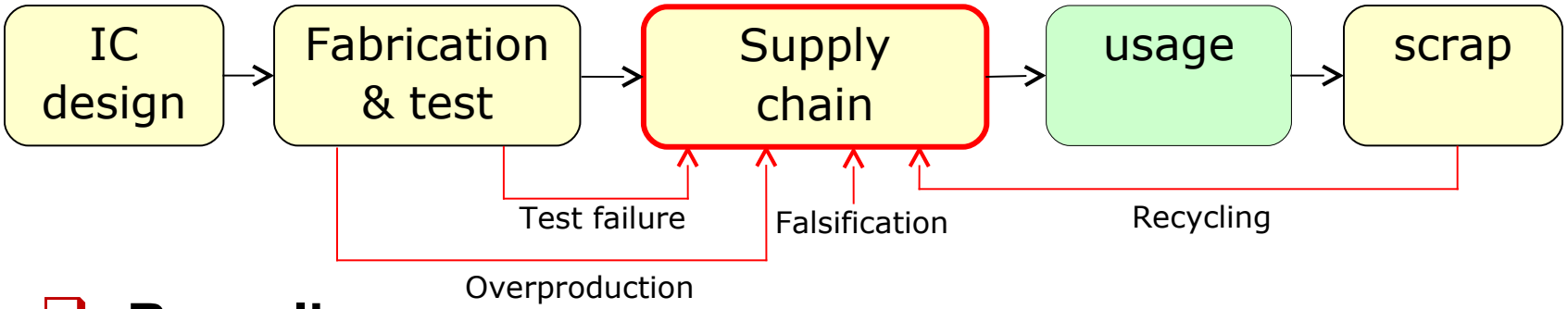
IC life cycle:



- **Counterfeiting types** [4]
  - The circuit is the **original** one but has been illegally manipulated
  - The circuit is **fake**

Counterfeiting with original circuit

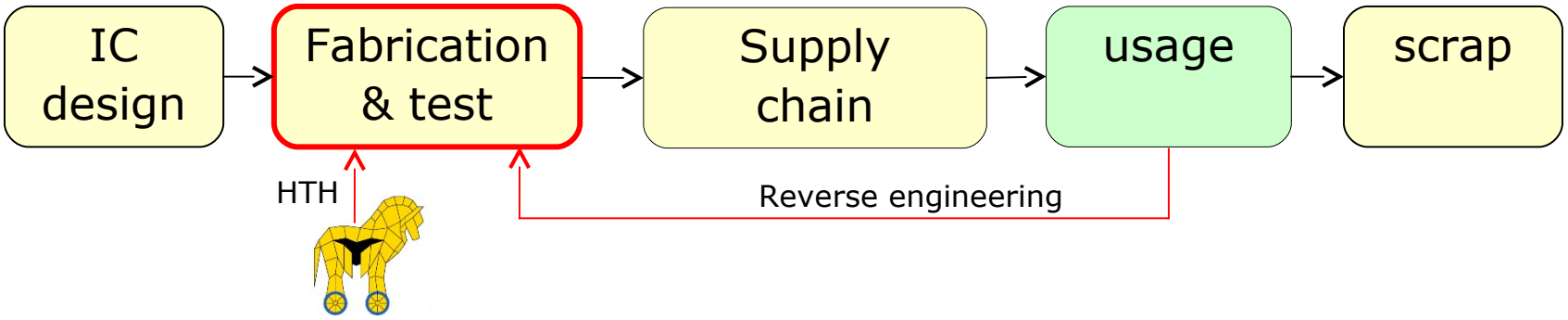
IC life cycle:



- **Recycling**
  - The circuit has been taken from old PCBs and remarked
- **Falsification**
  - The labeling, specification and certification are forged
- **Overproduction**
  - There is no legal contract for fabrication
- **With defects**
  - The component did not pass the tests

Counterfeiting with fake circuit

IC life cycle:



- **Cloning**
  - The circuit has been pirated by *reverse engineering* and redesigned identically
- **Hardware Trojan Horse (HTH)**
  - The circuit has been tampered at the fab stage and some extra logic called Hardware Trojan Horse has been added to spy or destroy it

How to protect from counterfeiting ?

- To work with **trusted partners**
  - Design House : to make ICs in trusted fabs
  - User : to buy ICs to trusted distributors
- To use **detection** techniques
  - For existing devices
  - For new devices with dedicated hardware
- To use **prevention** techniques
  - Only for new devices

Detection Methods [4]

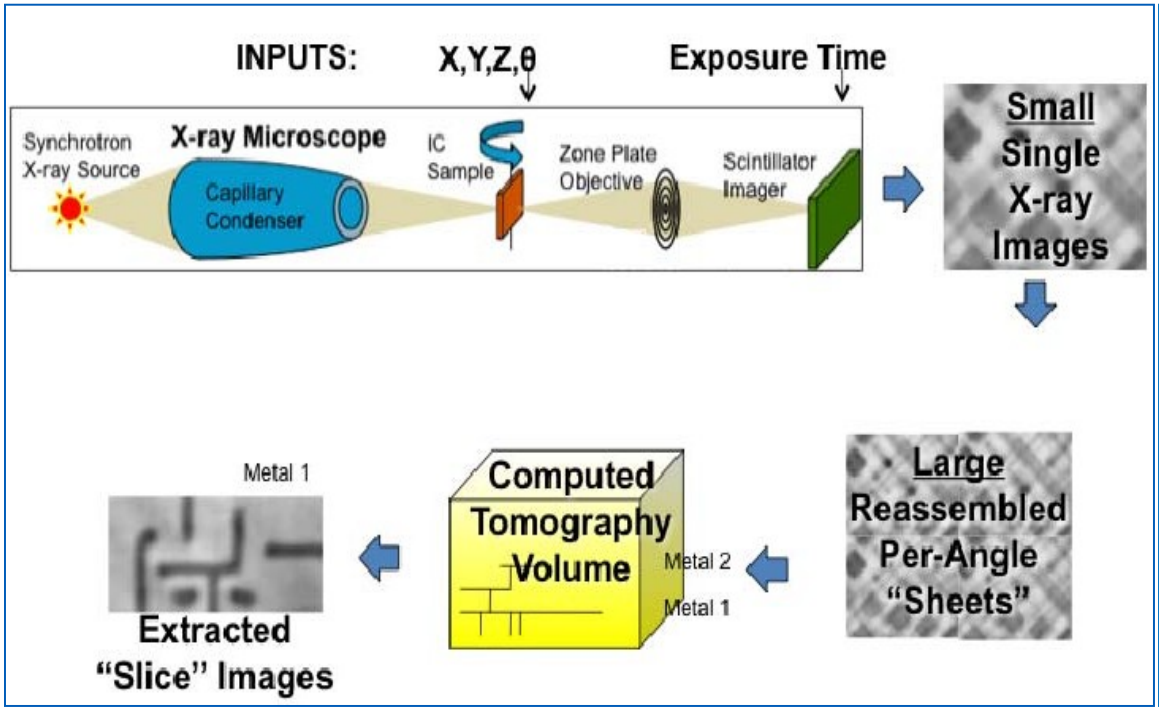
- 3 main types:
  - **Physical** analysis
    - Can be destructive
    - Only used to detect recycling and forged circuits
  - **Electrical** tests
  - **Aging** tests

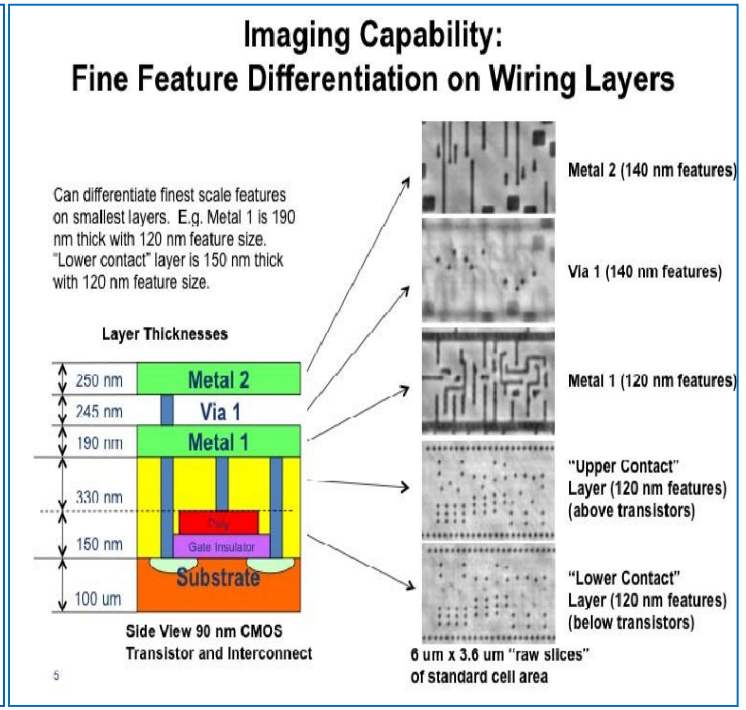
Detection: Physical analysis

- Imaging
  - Visual inspection
  - XRAY imaging
  - Scanning Acoustic Microscopy (ultrawave)
  - Scanning Electron Microscopy

- Material Analysis
  - XRAY fluorescence spectroscopy
  - IR spectroscopy (IR absorption)
  - THz spectroscopy (absorption in metal)

Example: X-ray Nanotomography[5]





Detection: Electrical tests

- Integrity tests
  - Scan chain to detect failures
- Parametric tests
  - DC and AC parameters in
  - In various environment
  - To detect abnormal offset
- Functionnal tests
  - To detect out of range ICs
- Burn-in test
  - Accelerate the **aging** and the failure occurence

Detection: Aging tests

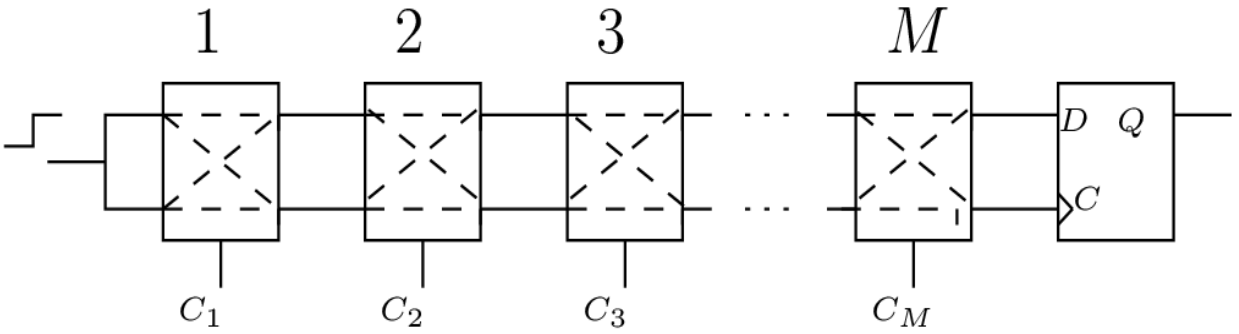
- Data analysis
  - Delay measurement which is very sensitive to aging
  - Machine Learning algorithms used to classify two sets of data:
    - New trusted devices and
    - unknown devices, presumably new
  - This method is impacted by process variation
- With internal sensors
  - CDIR (Combating Die and IC recycling)
  - Differential structure
    - Ring oscillator reference vs stressed
  - Usage time measurement
    - A clock counter is stored in NVM or OTP (antifuse)

Prevention: Hardware metering

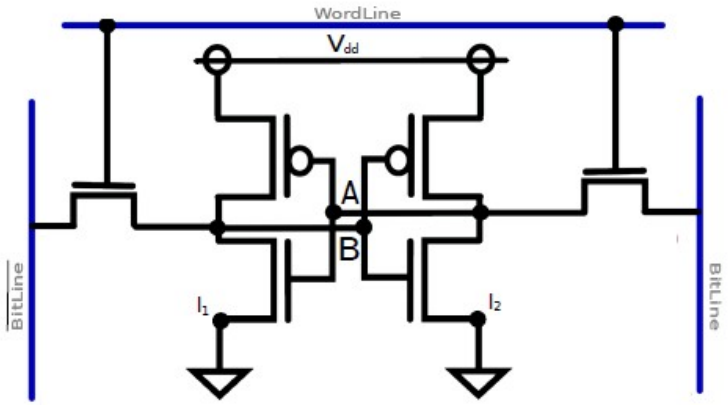
- **Passive** Hardware metering
  - Every circuit has its own ID, either in
  - Non Volatile memory : can be read or tampered
  - Physically Unclonable Function (PUF), which cannot be reverse engineered
  - An authentication protocol is build with the ID
- **Active** Hardware metering
  - The circuit is initially locked. It is unlocked only if the circuit is authenticated.

Prevention: PUF examples

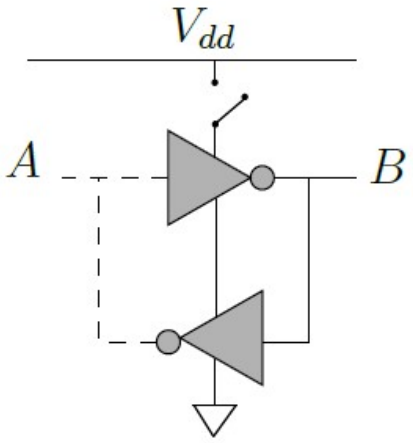
Arbiter PUF



SRAM PUF

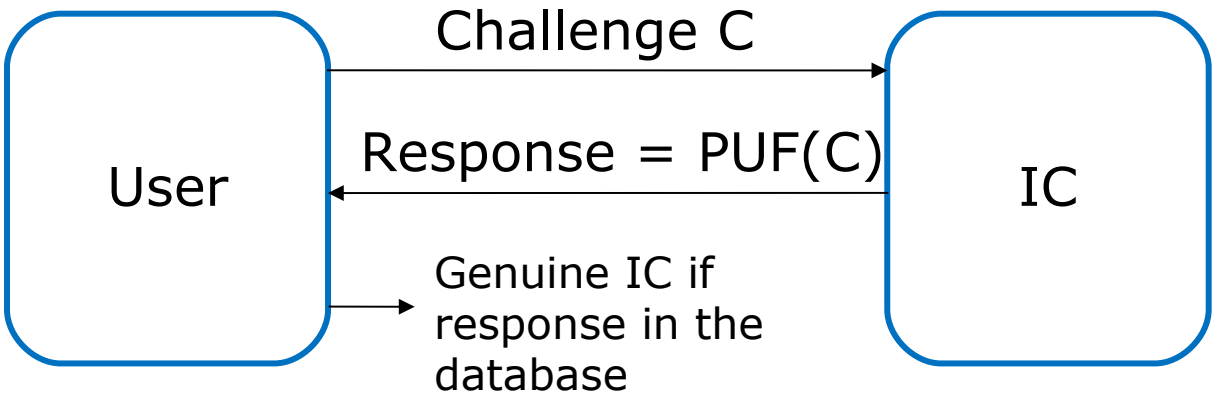


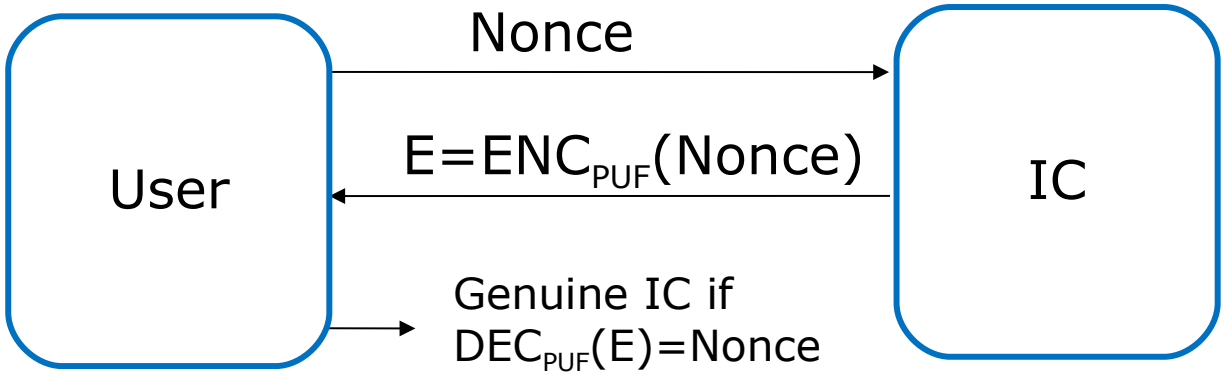
(a) SRAM cell CMOS circuit.



(b) SRAM cell logic circuit.

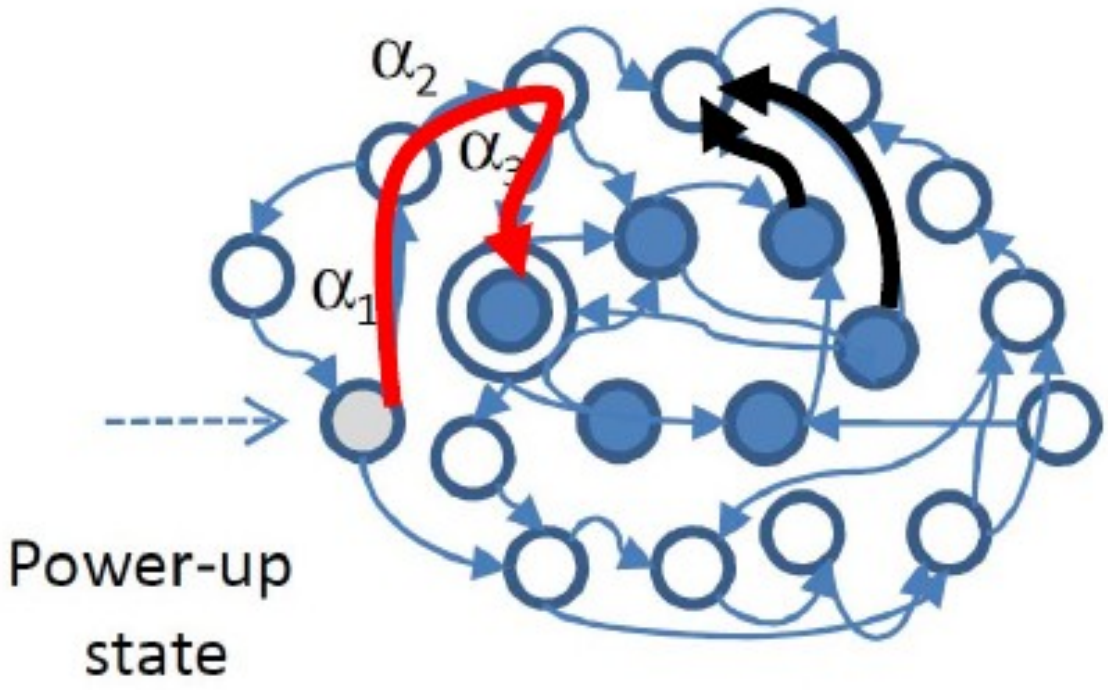
Prevention: PUF-based authentication





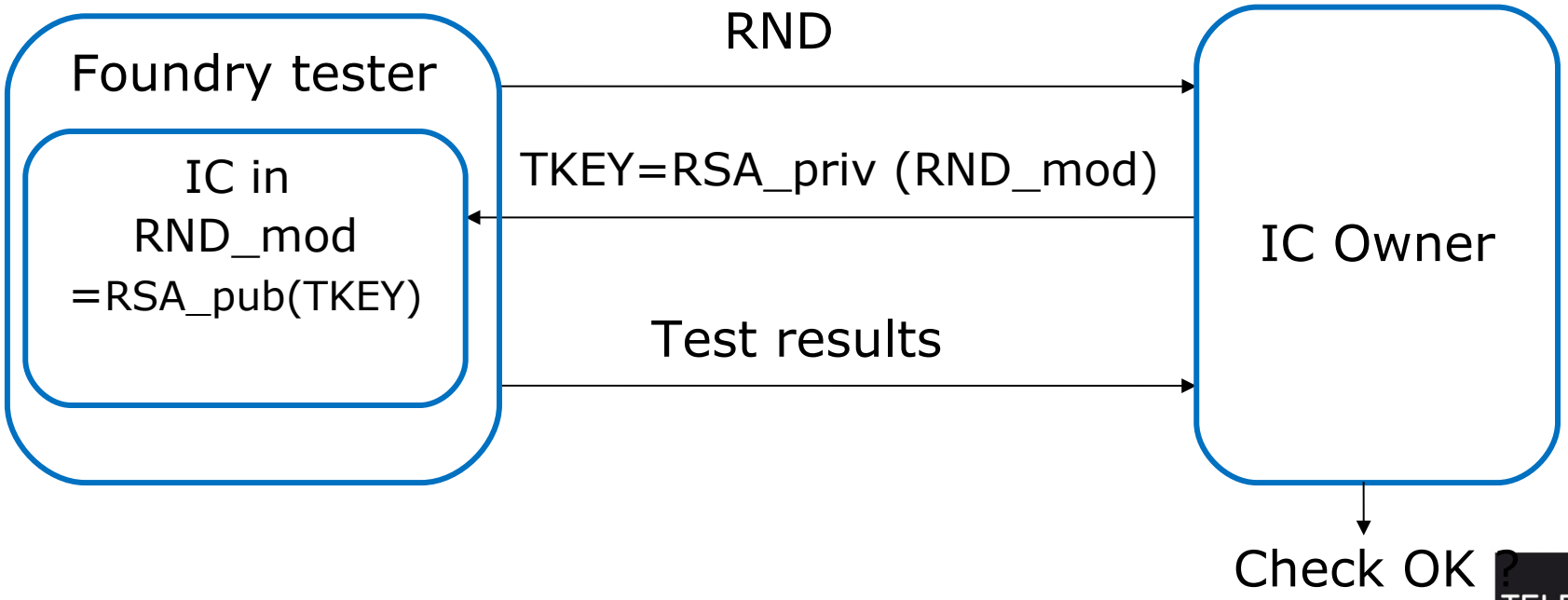
Prevention: locking mechanism [6]

- The state graph is locked at power-up, and unlocked with the correct sequence



Prevention: Secure Split Test [7]

- To secure the manufacturing test, hence counterfeiting with defects
- The test is done by using asymmetric crypto
- The IC owner unlocked the good IC with a master key



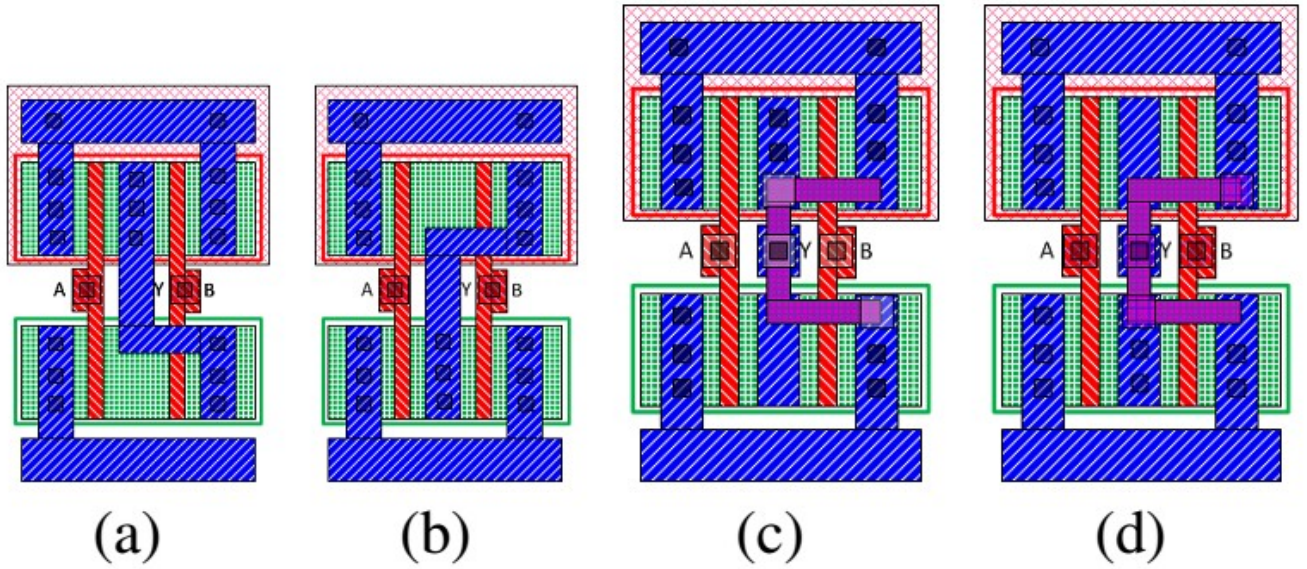
Prevention: Split manufacturing [8]

- The chip manufacturing is split into two steps.
- First step:
  - Done by any foundry, not necessarily trusted
  - In charge of the "**front end of line**": gates and first metal layers
- Second step:
  - Done by a trusted foundry
  - In charge of finishing the connections "**back end of line**"
- IARPA established a new program in 2011 based on split manufacturing: "Trusted Integrated Chips" [8]

Prevention: Watermarking [10]

- To secure IP (Intellectual Property) block inside an IC
- Many ways to insert the mark:
  - GDSII
    - Pattern specific to the design
  - FPGA
    - unused LUTs in the bitstream
  - HDL
    - Unused part of memory, or truth table combinations
  - Synthesis

Prevention: Camouflaging[11]



- Hiding of the cell layout to prevent reverse engineering by optical inspection
  - a: NAND, b: NOR
  - c: camouflaged NAND, d: camouflaged NOR

IC Counterfeiting protection efficiency

		Recycled	Forged	Over produced	Defective	Cloned	HTH
Detection	Physical tests	**	**				
	Electrical tests	**	**		*	*	*
	Aging tests	*	*				
Prevention	HW metering			*		*	
	Secure split test			**	**	*	
	Split manufacturing			*		*	*
	Watermarking					*	
	Camouflaging					*	

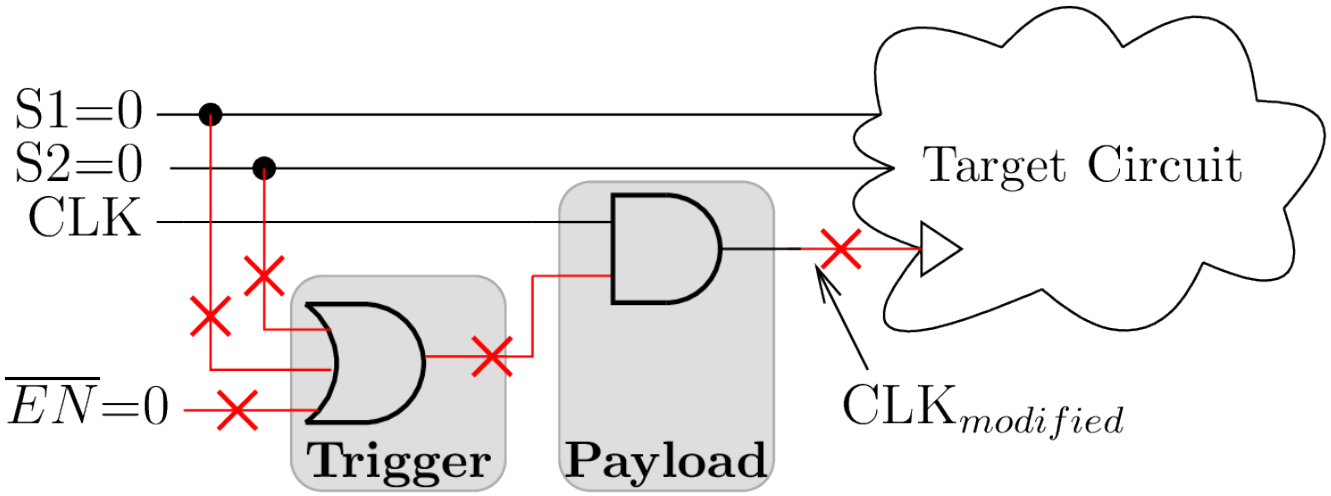
Outline

- IC Counterfeiting
  - Overview of the threat
  - Detection methods
  - Prevention methods
- Hardware Trojan Horses
  - Types
  - Detection methods
  - Prevention methods
- Conclusions

HTH: A powerful and pernicious threat

- HTH:
  - Insertion in an IC of Hardware unknown to the designer
  - Goal: spying, disturbing, destroying
  - Can be inserted at all the levels of the IC design chain
- It is not only an economic threat, it is also strategic
  - 2007 DARPA program "Trust in Ics"
  - 2011 IARPA program" Trusted Integrated Chips"[8] exloiting split manufacturing
- But it is also a weapon for the designer:
  - Backdoors

HTH Principle



- Two components:
  - TRIGGER: Reads and decode internal and rare state
  - PAYLOAD: Writes internal data
- HTH acts as a probing station, both passive (trigger) and active (payload), and is stealthy

HTH Payload examples[12]

- Kill switch
  - Simple payload, desastrous effect as Denial of Services
- Deteriorate the performances
  - Accelerate the aging, add extra delays
- Create leakages
  - Create an access to secret data, either by a functionnal channel or a side-channel
- Assist malwares
  - Exploits a hidden function. The HTH is called backdoor if the designer is the creator.

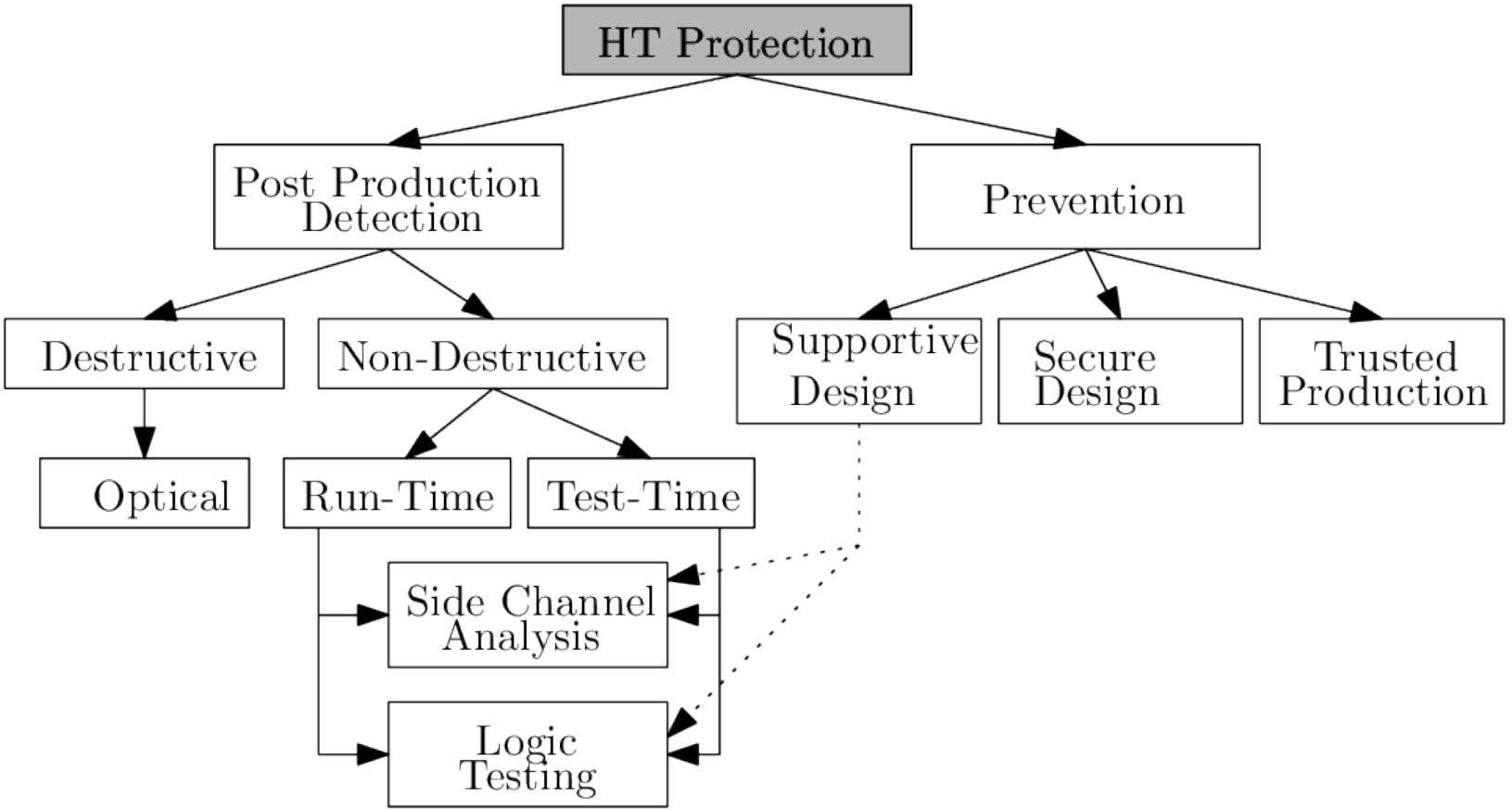
HTH Triggering examples

- Combinatorial
  - Decoding of rare event from multiple nodes
    - trigger = f(nodes)
  - Use significant number of gates
- Sequential
  - Decoding of a rare event from sequential variables
    - Trigger = f(nodes, time)
    - Less nodes but a few flip-flops
- Analog
  - Use internal sensors and external parameters
    - Example: Trigger temperature > threshold
  - Need few gates

HTH Taxonomy [13]

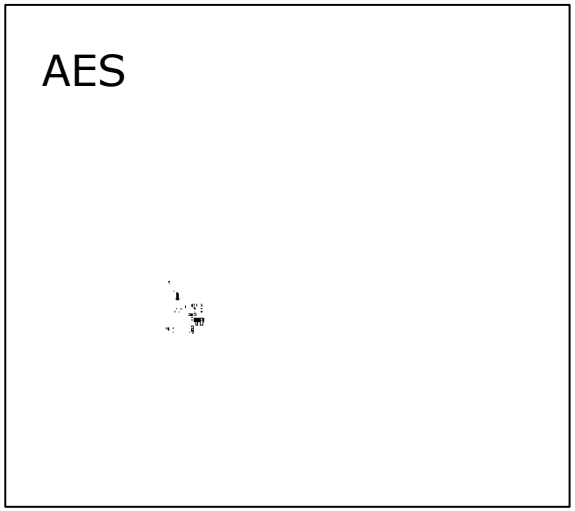
- **Insertion** stage
  - Specification, design, fabrication, test, assembly
- **Abstraction** level
  - RTL, gate, layout, physical,
- **Trigger** type
  - Combinatorial, sequential, analog
- **Payload** type
  - New behavior, less performance, leakages, DoS
- **Physical** characteristics
  - Size, distribution, parametric, functionnal, same layout

HTH protection overview [14]

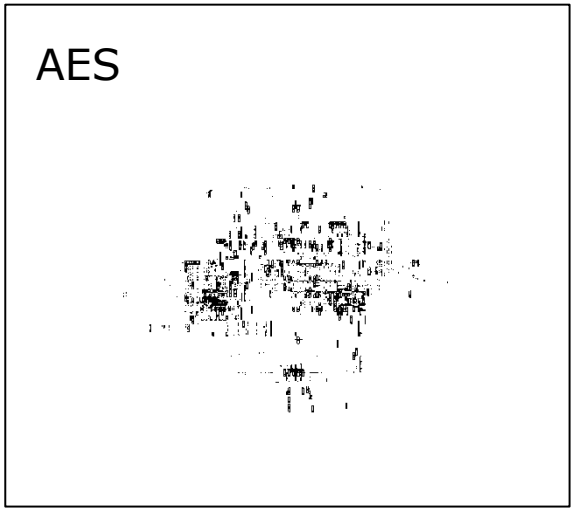


HTH detection by optical method[15]

- Needs a GDSII golden model



Trojan size = 1 AND gate



Trojan size = 128 AND gate

- Comparison between an original GDSII and a trojaned IC with a ×150 lens confocal microscope

HTH detection by optical method

- Cross correlation between the original AES layout and an affected AES layout

		Hardware Trojan size (Nb of AND gates)							
		1	2	4	8	16	32	64	128
Core utilization rate	50%	0.9991	0.9972	0.9981	0.9950	0.9933	0.9918	0.9815	0.9668
	60%	0.9987	0.9968	0.9959	0.9955	0.9944	0.9893	0.9788	0.9670
	70%	0.9989	0.9981	0.9918	0.9941	0.9881	0.9850	0.9594	0.9067
	80%	0.9999	0.9965	0.9898	0.9957	0.9780	0.9711	0.8970	0.8509
	90%	0.9988	0.9990	0.9983	0.9962	0.9832	0.9572	0.8858	0.4010
	95%	0.9997	0.9984	0.9980	0.9889	0.9589	0.9115	0.8824	0.8202
	99%	0.9917	0.938	0.9714	0.9527	0.3798	NC	NC	NC

In black: ECO routing

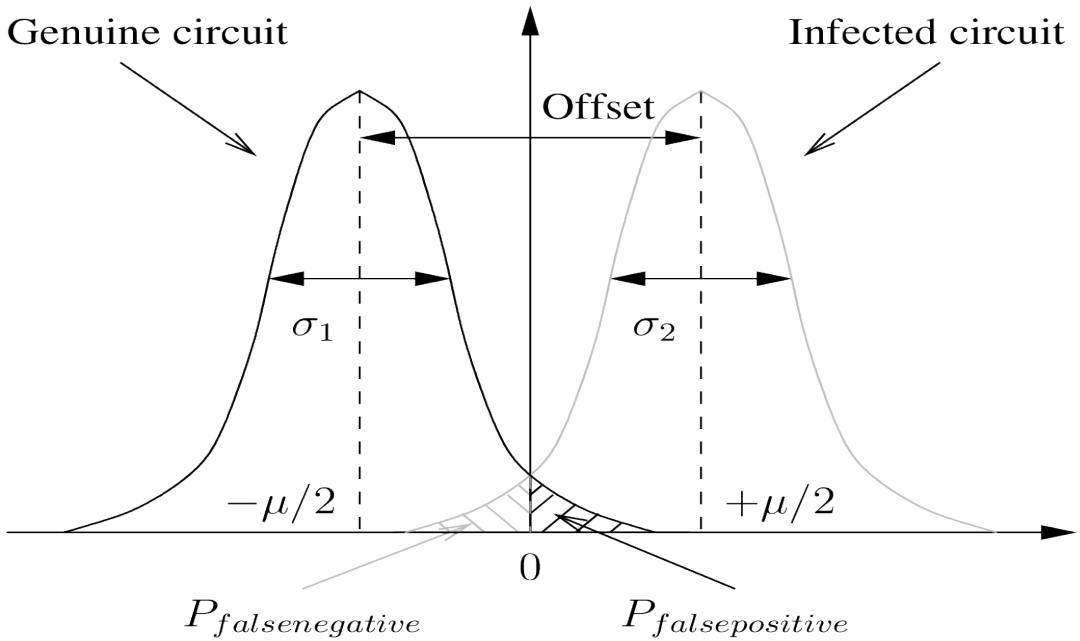
- Trojan almost impossible to insert without changing the layout, if occupancy rate > 80%

HTH detection at test time

- Logic Testing
  - To search the triggering of the Trojan
  - Need exhaustive search of a rare event => impossible
  - Rather use statistical approaches as MERO[15]
  - Or add HW to avoid rare event
- Side Channel
  - To detect the resources of the Trojan
  - By measuring:
    - the Current (IDDQ, IDDT)
    - the EM field
    - the propagation delays
  - Very sensitive to noise and process variance

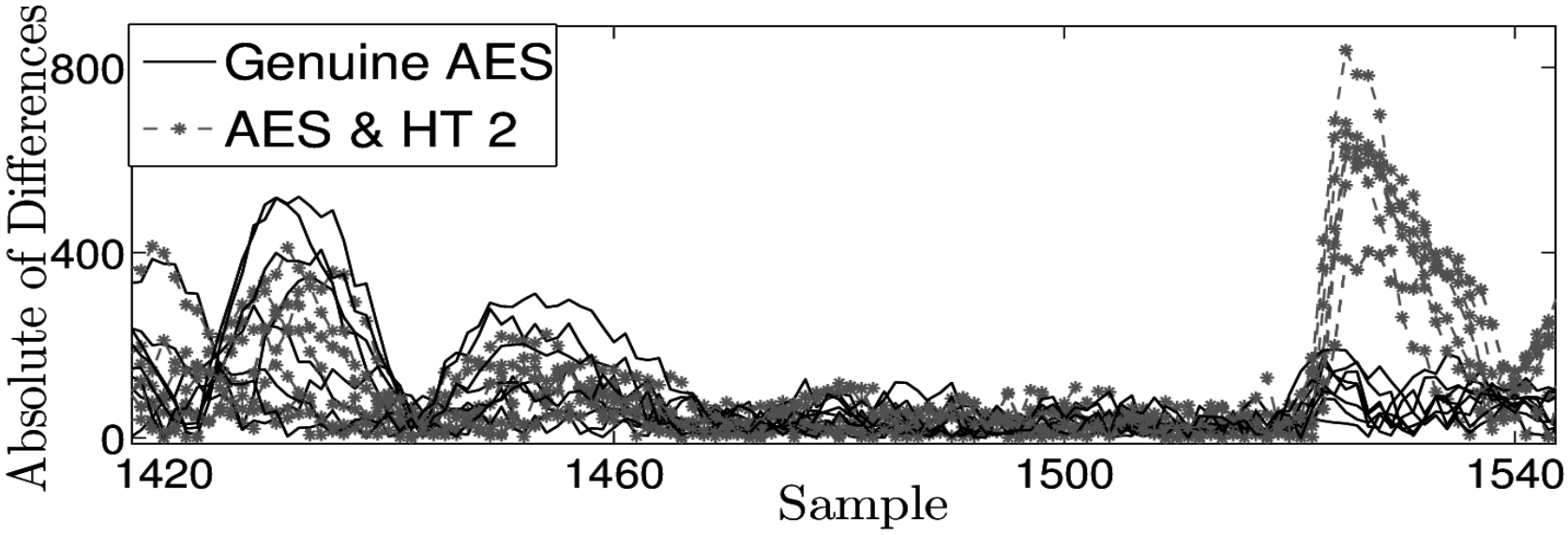
HTH detection with side-channel[17]

- Needs a golden model of the "activity"
- Measurement of local EM field with RF probes
- Impact of noise => Probability of detection



HTH detection with side-channel

- Impact of process variance



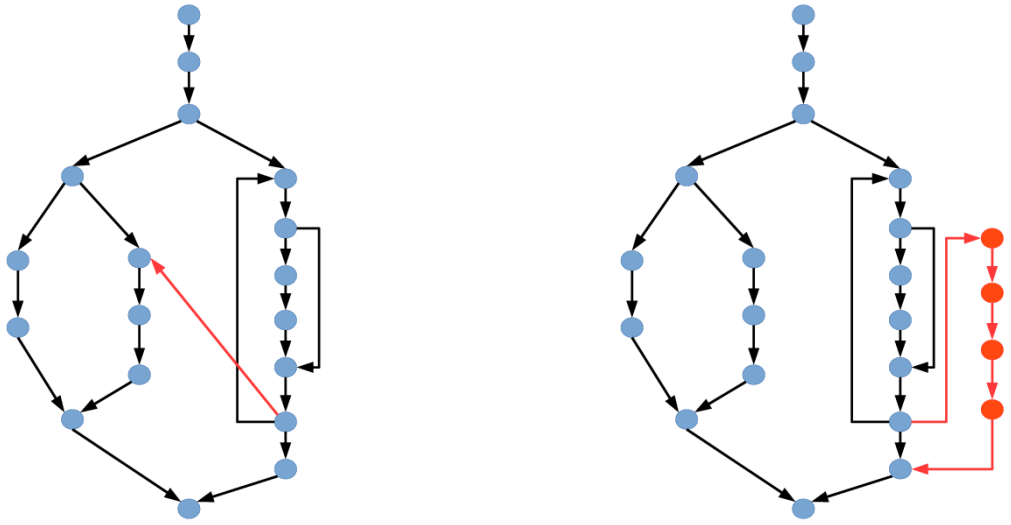
- HTH of different sizes: HTH greater than 1% can be detected with a false negative rate of 0.017%.

HTH detection at run time

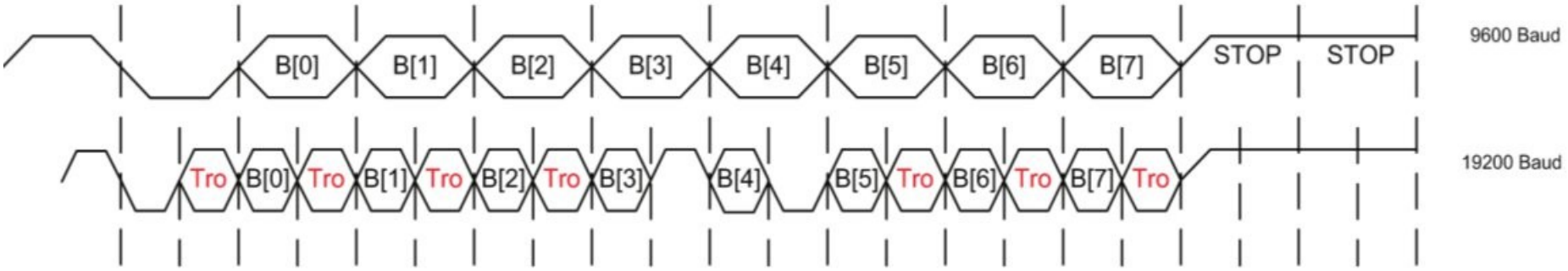
- Techniques to check the integrity in real time
- Can take advantage of SEU and attack detection techniques :
  - Error correction codes
  - Control Flow Integrity (processors)
  - Hardware Assertions checkers
  - Real time security monitor

HTH detection by flow integrity check[18]

- The processor control flow can be tampered by HTH and/or malwares
- Prevention can check the integrity of basic blocks and unexpected jumps
  - Example: Use golden tables of basic blocks CRC and jump tables



HTH detection at run time: assertions[19]



- Example: The HTH outputs a secret key with on the UART channel by doubling the Baudrate
- Property to check by Hardware:
  - The serial bits have to be stable during a fixed period.
  - If the baudrate changes, the assertion fails

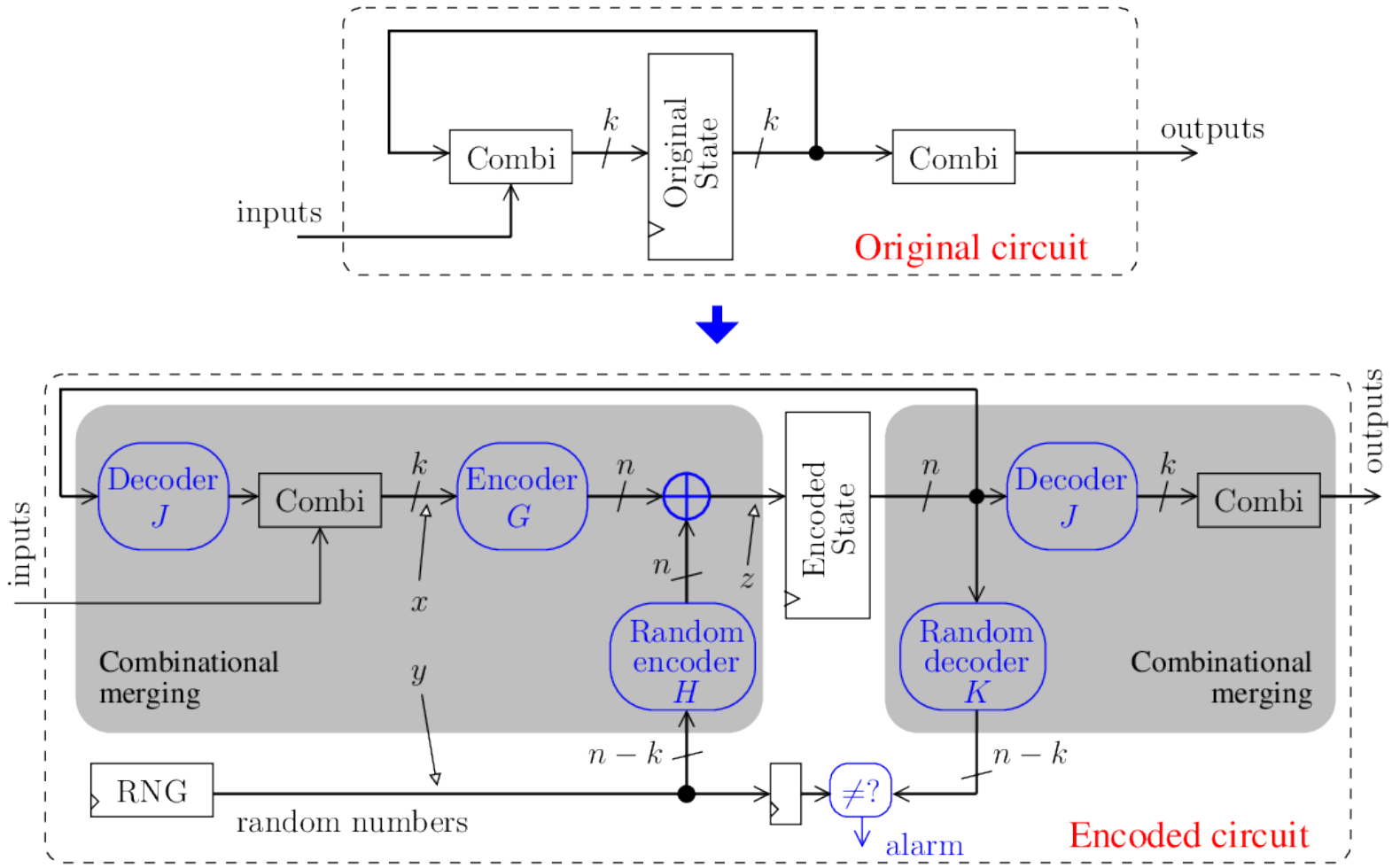
HTH prevention

- **Split manufacturing**
  - Use a "root of trust" with two steps: Front-end of line, and Back end of line
- To use the **layout-filler**
  - No more places to insert HTH on GDSII
- To **avoid rare events** during test time
- **Obfuscation**
  - To obfuscate the state transitions by keys
    - Active **Hardware metering**
  - To obfuscate by error correcting codes (**ECC**)
    - Mask the signals with random variables

Encoding the circuit [20]

- Principle:
  - The HTH has two parts:
    - **probing** (trigger) and **fault** injection (payload)
  - The **registers** are the most easy cells to detect, thus the most easy to probe for Trojan insertion
  - The **sequential variables** in registers are encoded by Linear Complementary Dual Codes (**LCD**)
  - The Dual code allows the designer to use random variables to mask the real computation
- Protection also effective against:
  - Probing attacks
  - Fault attacks
  - Side-Channel Attacks

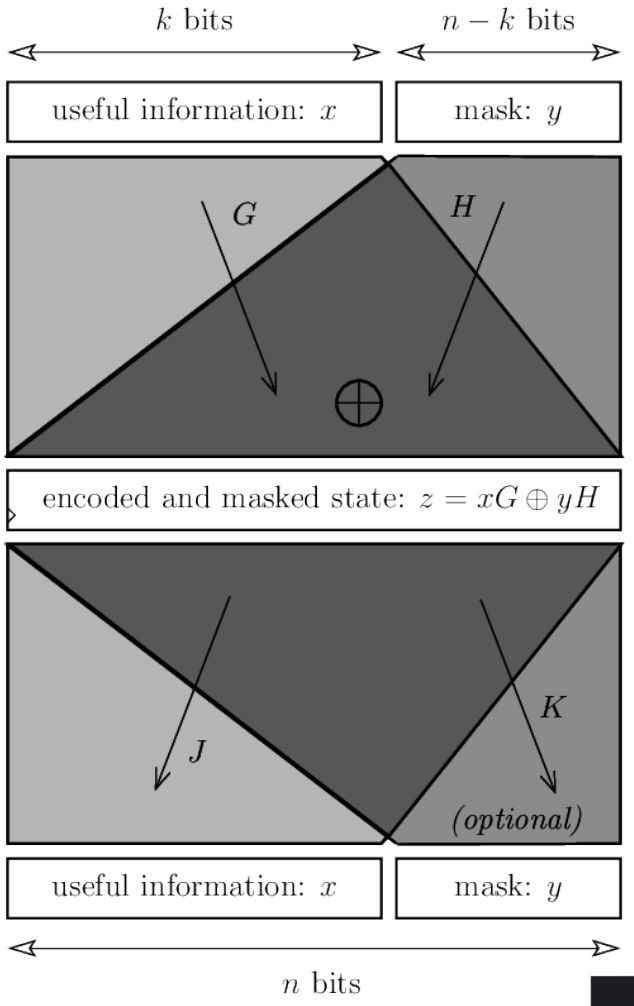
Encoding the circuit: Architecture



Encoding the circuit: Methods

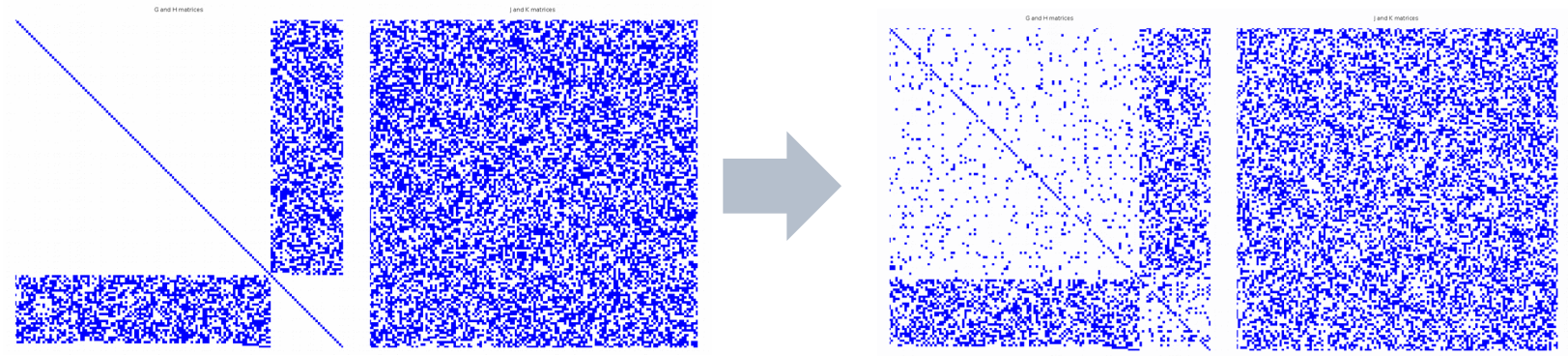
- G encodes k bits
- H is the dual of G
  - ($GH^T=0$)
- Encoding:
  - Z=xG xor yH
  - X=information
  - Y=random variable
- Decoding*
  - $J=G^T(GG^T)^{-1}$
  - $K=H^T(HH^T)^{-1}$

* Moore-Penrose pseudo-inverse



Encoding the circuit: Security Proof

- The code [n,k,d] has a proven security of d:
  - The HTH **trigger** is inefficient with less than d probes
  - The HTH **payload** is inefficient if it modifies less than d nets
- Complexity
  - Choose low density codes to encode and decode



Encoding complexity

Table 8.6 – Synthesis results of encoded circuit method, and security parameters for the SIMON co-processor.

IC (Code)	**Gates**	**Area (μm^2)**	n	k	$d_{\textbf{Trigger}}$	$d_{\textbf{Payload}}$
Original ([109,109,1])	300	1919	109	109	1	1
Encoded ([110,109,2,1])	560	3567	110	109	2	1
Encoded ([140,109,10,6])	3107	20239	140	109	10	6
Encoded ([123,109,5,3])	2348	15249	123	109	5	3

Conclusions

- Methods for counterfeiting and HTH insertions are sophisticated and increasing.
- Many protections:
  - But need resources:
    - Tools and methods for detection
    - Extra Silicon and methods for prevention
    - Split foundries
  - The optimal solution is still a challenge
    - Combination of techniques
    - With reduced complexity to get higher detection or avoidance rate
  - But very few inputs from the industrials

Key References

1. http://www.semiconductors.org/document_library_sia/anti_counterfeiting/sia_whitepaper_winning_the_battle_against_counterfeit_semiconductor_products/
2. http://www.armed-services.senate.gov/download/inquiry-into-counterfeit-electronic-parts-in-the-department-of-defense-supply-chain
3. http://www.zdnet.com/article/counterfeit-chips-a-169-billion-tech-supply-chain-headache/
4. Guin, U., Huang, K., DiMase, D., Carulli, J. M., Tehranipoor, M., & Makris, Y. (2014). Counterfeit integrated circuits: a rising threat in the global semiconductor supply chain. Proceedings of the IEEE, 102(8), 1207-1228.
5. Bajura, M., Boverman, G., Tan, J., Wagenbreth, G., Rogers, C. M., Feser, M., ... & Reynolds, P. (2011, March). Imaging Integrated Circuits with X-ray Microscopy. In Proceedings of the 36th GOMACTech Conference.
6. Alkabani, Y., & Koushanfar, F. (2007, August). Active Hardware Metering for Intellectual Property Protection and Security. In USENIX Security (pp. 291-306).
7. Contreras, G. K., Rahman, M. T., & Tehranipoor, M. (2013, October). Secure split-test for preventing IC piracy by untrusted foundry and assembly. In Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2013 IEEE International Symposium on (pp. 196-203). IEEE.
8. Jarvis, R. W., & McIntyre, M. G. (2007). U.S. Patent No. 7,195,931. Washington, DC: U.S. Patent and Trademark Office.
9. http://www.iarpa.gov/index.php/research-programs/tic
10. Charben, E. (1998, May). Hierarchical watermarking in IC design. In Proceedings of the IEEE Custom Integrated Circuits Conference (pp. 295-298). IEEE.
11. Rajendran, J., Sam, M., Sinanoglu, O., & Karri, R. (2013, November). Security analysis of integrated circuit camouflaging. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (pp. 709-720). ACM.
12. https://www.trust-hub.org
13. Rajendran, J., Gavas, E., Jimenez, J., Padman, V., & Karri, R. (2010, May). Towards a comprehensive and systematic classification of hardware trojans. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on* (pp. 1871-1874). IEEE.
14. Francq, J., & Frick, F. (2015, March). Introduction to hardware trojan detection methods. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition* (pp. 770-775). EDA Consortium.
15. Bhasin, S., Danger, J. L., Guilley, S., Ngo, X. T., & Sauvage, L. (2013, August). Hardware trojan horses in cryptographic ip cores. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2013 Workshop on* (pp. 15-29). IEEE.
16. Chakraborty, R. S., Wolff, F., Paul, S., Papachristou, C., & Bhunia, S. (2009). MERO: A statistical approach for hardware Trojan detection. In *Cryptographic Hardware and Embedded Systems-CHES 2009* (pp. 396-410). Springer Berlin Heidelberg.
17. Ngo, X. T., Najm, Z., Guilley, S., Bhasin, S., & Danger, J. L. (2014). Method Taking into Account Process Dispersion to Detect Hardware Trojan Horse by Side-Channel. *Proc. Security Proofs for Embedded Systems--PROOFS*.
18. Danger, J. L., Guilley, S., Porteboeuf, T., Praden, F., & Timbert, M. (2014, December). HCODE: Hardware-Enhanced Real-Time CFI. In *Proceedings of the 4th Program Protection and Reverse Engineering Workshop* (p. 6). ACM.
19. Ngo, X. T., Danger, J. L., Guilley, S., Najm, Z., & Emery, O. (2015, August). Hardware property checker for run-time Hardware Trojan detection. In *Circuit Theory and Design (ECCTD), 2015 European Conference on* (pp. 1-4). IEEE.
20. Ngo, X. T., Guilley, S., Bhasin, S., Danger, J. L., & Najm, Z. (2014, October). Encoding the state of integrated circuits: a proactive and reactive protection against hardware Trojans horses. In *Proceedings of the 9th Workshop on Embedded Systems Security* (p. 7). ACM.