

**The Fourth International Conference on Dependability - DEPEND 2011**  
**August 21, 2011 - French Riviera, Nice/Saint Laurent du Var, France**

---

# **Taking Care of Security in Hardware Design**

**Régis Leveugle**

**Professor, Grenoble INP / TIMA, France**

**Regis.Leveugle@imag.fr**



**TIMA Laboratory**  
**46 av. Félix Viallet - 38031 Grenoble Cedex - France**

**R. Leveugle**

# Why being concerned by chip-level security ?

---

- **Because security is an increasing societal and economics concern in the "information society",**
  - ◆ **Mastering some kind of information is now the key in most situations**
  - ◆ **Need for availability, integrity, authenticity of services**
  - ◆ **Ensuring privacy is required in spite of all the information flows**
  - ◆ **...**
  
- **... More and more deeply inserted at the heart of many of our everyday-life applications,**
  
- **... Implemented by "embedded systems", that are in most cases integrated systems (with on-chip cryptography),**
  
- **... Thus very dependent on potential flaws at the chip level.**



# Security flaws at chip level ??

---



- **Yes, many examples ! Not only Internet viruses or big infrastructure weaknesses ...**
  
- **A recent one ?**
  - "T card has security flaw, says researcher", by Hiawatha Bray**  
**The Boston Globe (boston.com), March 6, 2008**
  
  - ◆ **A computer science student at the University of Virginia asserts that he has found a security flaw in the technology behind the Massachusetts Bay Transportation Authority's CharlieCard system (based on the MiFare Classic RFID chip by NXP Semiconductors). The company spent \$192 million to introduce the CharlieCard in 2006. The system replaced cash and tokens.**
  
  - ◆ **Such a breakthrough could be used to make counterfeit copies of the cards (sold in the underworld), allowing commuters to pay for bus and subway rides.**
  
  - ◆ **It is also the chip used in London's subway system.**

# This should be very complex and expensive !!

---



## □ Complex ?

- ◆ "A press release issued by the University of Virginia said the research team obtained the same kind of chip, then used abrasives to scrape away the chip layer by layer. By examining the chip circuitry, they were able to figure out the encryption algorithm it uses and found weaknesses that made it easy to break. Next, the team was able to use commercially available RFID readers to capture data from any RFID-equipped cards that came within range. They could then decrypt the data on those cards and copy them."
- ◆ Not for everybody ... but far from impossible ...

## □ Expensive ?

- ◆ "Nohl said that his team needed only about \$1,000 worth of equipment to dismantle the chip and crack the code."
- ◆ Techniques are in some cases still cheaper ...

# Few comments on this case

---

- **Several security layers => One breach does not imply the full system failure**
  
- **Risk of (large) financial losses in this case for the provider – Many similar cases (cash from ATMs, pay-per-view TV access, ...)**
  
- **Consequences may be more dangerous in other cases (unauthorized access by a terrorist or a spy in a critical area ...)**
  
- **Chip-level security is a concern !**
  - ◆ **Attacks – need to understand possible chip-level weaknesses (even without design error)**
  - ◆ **Counter-measures – need to develop protections w.r.t. those attacks**



# Outline

---

- **Secured circuits: design constraints, qualification, common criteria**
  
- **Circuit-level attacks (based on hardware implementation)**
  - ◆ Types of attacks – taxonomy and exploitation examples
  - ◆ Error models – Cell-based design and SRAM-based FPGAs
  
- **Design for security (hardening by design): examples**
  - ◆ w.r.t. invasive attacks
  - ◆ w.r.t. side-channel attacks
  - ◆ w.r.t. fault-based attacks
  
- **Influence of design – e.g. synchronous vs. asynchronous design**

**Examples are given on well known algorithms (AES, RSA ...)**

---



---

# Design and qualification of secured circuits



# Security definition

---

- **Security = one attribute of dependability**
  
- **Security is the concurrent existence of several pillars**
  - ◆ **Availability** (but for authorized users only => **Confidentiality**),
  - ◆ **Integrity** (absence of improper system alterations) with "improper" meaning here "unauthorized",
  - ◆ **Authenticity** (information comes from trusted source)
  - ◆ **Resilience** (or robustness w.r.t. attacks against previous pillars + limiting the potential damage in case of successful attack)
  
- **Often linked to the use of cryptography (or steganography)**  
=> most examples will be given on crypto-processors or cryptographic accelerators
  
- **System-level security policies will not be discussed here**





# Typical examples of (secure) embedded systems ...

---



**Note: the security level of course depends on the application**

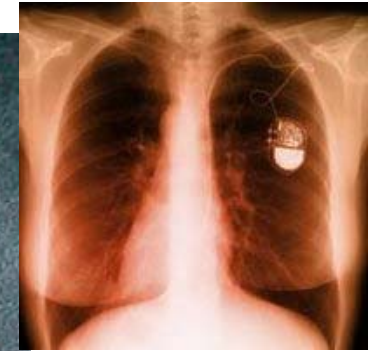
---

# Other example ... implanted devices

From ...



... to ...



Today, a pace-maker is a small computer able to analyze the heart electrical signals that can be tested and reprogrammed by telemetry (contactless communications)

Dependability constraints (with minimum power consumption !):

**Reliability:** difficult replacement !

**Safety:** EM perturbations (e.g. iPod ...)

<http://www.techshout.com/ipod/2007/11/ipods-can-cause-pacemakers-to-malfunction-study/>

**Security:** malicious activation/deprogramming by a hacker

<http://www.newscientist.com/blog/technology/2008/03/death-by-radio-waves-hacking-pacemaker.html>

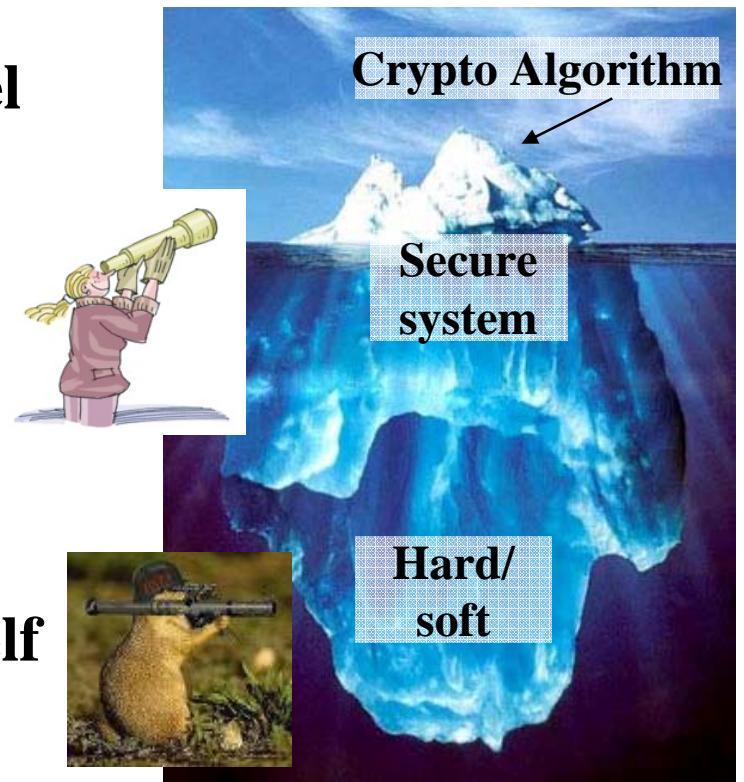
# Tampering secure circuits

---

- ❑ **A secure circuit contains secret data (e.g. a secret cryptographic key)**
- ❑ **Knowing the secret grants unauthorized privileges (access, message interception, ...)**
- ❑ **May allow in some cases to clone a device (e.g. pay-per-view TV decoder ... or a bus/subway card ...)**

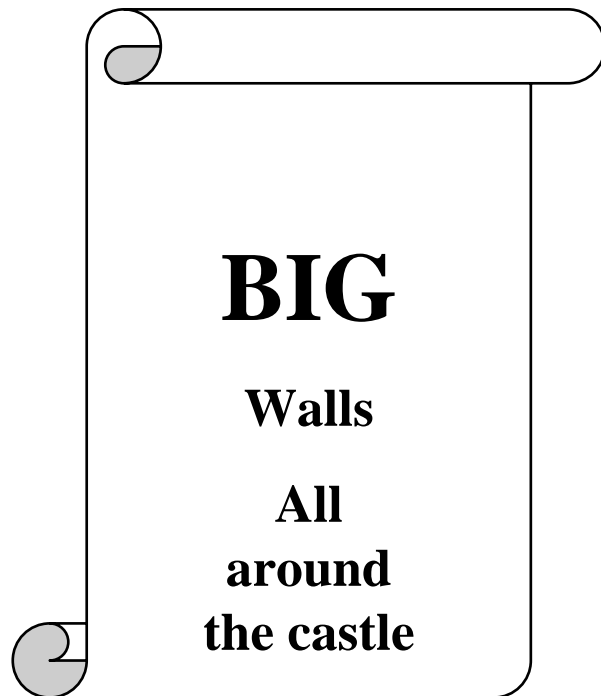
# Attacks vs. classical cryptanalysis

- **Classical cryptanalysis mainly aims at finding**
  - ◆ **Algorithm flaws (mathematical analysis)**
  - ◆ **Practical limits in exhaustive search (brute-force attacks)**
  
- **Careful study of the mathematical model describing a cryptosystem, yielding properties revealing information about the secret**
  
- **General, sometimes very theoretical**
  
- **Attacks target the implementation characteristics, not the algorithm by itself**
  
- **Specific, much more practical**

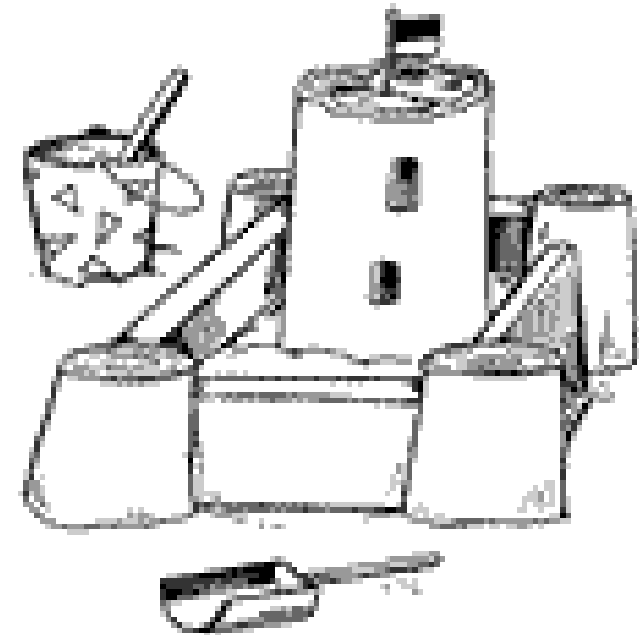


# Algorithm vs. implementation

---



**Algorithm**



**Implementation**

**Hardware-level security is at the heart of system security**

---

# Secure product validation and qualification

---

## □ Several steps

- ◆ Design time (simulations, proofs, ...)
- ◆ Final product samples (official qualification – national certified laboratories or evaluation centers)

## □ Multi-aspect evaluation

- ◆ Product robustness – attacks
- ◆ Source code analysis (hardware and software)
- ◆ Design/development process (methods, tools, controlled environment, ...)

## □ International reference: Common Criteria (among other standards)

- ◆ With respect to specified threats (not an absolute qualification)
- ◆ White box evaluation: not representative of real hacker attacks



# Common Criteria (CC)

---

- **Common Criteria for Information Technology Security Evaluation**
- **ISO 15408**
  
- **Framework for specification of evaluation**
  - ◆ **Security target (assets, threats => security functions and requirements)**
  - ◆ **Protection Profile (PP)**
  - ◆ **Evaluation Assurance Level (EAL 1-7)**
  
- **Aims at evaluating:**
  - ◆ **Product conformance (w.r.t. claims/technical datasheet)**
  - ◆ **Robustness of implemented mechanisms**
  - ◆ **Confidence in the development process**

# Attack levels (Robustness levels)

---

- **CC version 3.1 : 5 levels**
  - ◆ **Public vulnerability (AVA\_VAN.1)**
  - ◆ **Elementary attack (AVA\_VAN.2)**
  - ◆ **Re-enforced elementary attack (AVA\_VAN.3)**
  - ◆ **Medium attack level (AVA\_VAN.4)**
  - ◆ **High attack level (AVA\_VAN.5)**
  
- **Assumptions about the type of attacker (knowledge, equipment)**



# Evaluation grades

---

- **Estimation of:**
  - ◆ **Required equipment**
  - ◆ **Required experimental duration**
  - ◆ **Required knowledge on the product (e.g. black vs. white box attack)**
  - ◆ **Required expertise**
  - ◆ **Required opportunity**
  
- **Sum of grades => intervals compatible with the 5 robustness levels**

# 7 fixed Evaluation Assurance Levels

---

- **EAL1 – Functionality Tested**
  - **EAL2 – Structurally Tested**
  - **EAL3 – Methodically tested and checked**
  - **EAL4 – Methodically Designed, Tested, and Reviewed**
  - **EAL5 – Semiformally Designed and Tested**
  - **EAL6 – Semiformally Verified Design and Tested**
  - **EAL7 – Formally Verified Design and Tested**
- } **High levels**

... or "How well we can underpin that the attacks can be countered ?"



# EAL4

---

- **White box evaluation (full visibility on the product)**
  
- **Requires for the security functions:**
  - ◆ **Complete specifications**
  - ◆ **Rigorous design**
  - ◆ **Audit of implementation sample (source code – C, VHDL, ...)**
  - ◆ **Complete user and administration guides**
  - ◆ **Integration and validation tests (test plan, results)**
  - ◆ **Development process mastering and control**
  - ◆ **AVA\_VAN.3 robustness level**
  
- **24 assurance components – 110 requirements**

# EAL5

---

- **Semi-formal specifications**
  
- **Requires for the security functions:**
  - ◆ Complete **semi-formal** specifications
  - ◆ **Semi-formal preliminary design**
  - ◆ **Modular detailed design of security functions**
  - ◆ **Audit of implementation part**
  - ◆ **Use of standard coding approaches**
  - ◆ **Complete user and administration guides**
  - ◆ **Unitary, integration and validation tests**
  - ◆ **Development process mastering and control**
  - ◆ **AVA\_VAN.4** robustness level
  
- **25 assurance components – 118 requirements**



# EAL6

---

- **Semi-formal design**
  
- **Requires for the security functions:**
  - ◆ Complete semi-formal specifications
  - ◆ **Formal specification of the implemented security policies**
  - ◆ Semi-formal preliminary design
  - ◆ **Complete modular** detailed design
  - ◆ Audit of **complete** implementation
  - ◆ Complete user and administration guides
  - ◆ Unitary, integration and validation tests
  - ◆ Development process mastering and control (**including COTS**)
  - ◆ **AVA\_VAN.5** robustness level
  
- **26 assurance components – 122 requirements**



# EAL7

---

- **Formal specification**
- **Requires for the security functions:**
  - ◆ **Formal specifications**
  - ◆ **Formal specification of the implemented security policies**
  - ◆ **Formal preliminary design**
  - ◆ **Complete semi-formal detailed design**
  - ◆ **Audit of complete implementation**
  - ◆ **Complete user and administration guides**
  - ◆ **Complete tests of source code with coverage demonstration**
  - ◆ **Complete tests independent from the evaluator**
  - ◆ **Development process mastering and control (including COTS, quality measure)**
  - ◆ **AVA\_VAN.5 robustness level**
- **26 assurance components – 126 requirements**



# Evaluation limitations ...

---

- **Few products evaluated above EAL5 (about 50 EAL5 worldwide in Q1 2008, mainly smartcards – many others on-going – and 2 products qualified EAL6, EAL7)**
  
- **Most common products not evaluated (especially for software products)**
  - ◆ **Evaluation takes a long time (even with incremental process)**
  - ◆ **Modern patch/upgrade process hardly compatible with evaluation**
  - ◆ **E.g., Windows versions**
    - **Do you use EAL4 certified “Microsoft Windows XP, Professional; SP 2 (hotfixes 896423, 899587, 899588, 896422, 890859, 873333, 885250, 888302, 885835, and 907865)”**
    - **Or the most current release, with fewer known vulnerabilities ?**
  
- **Similar problem increasing for hardware products (reconfigurable chips)**



# Case of SoCs and consumer market

---

- **Increasing need for security in (complex) consumer devices**
- **Historically few concerns about security (open systems) but**
  - ◆ **Automotive**
  - ◆ **Set-top-boxes, mobile (smart)phones**
  - ◆ **Digital camera, digital TV, multimedia, DVD player, game stations**
  - ◆ **...**
- **Dedicated security modules are often not an adequate solution**
- **Need for SoC (and MPSoC) robustness against HW & SW attacks (at low cost)**
- **But consumer MPSoCs are not smartcards (high complexity ICs, small market window ...) => CC certification usually not used**





---

# Hardware-based attacks



---

# Types of attacks, exploitation examples



# Types of Implementation (HW) based attacks

---

- **Observation-based (passive, side channels)**
- **Perturbation-based (active, fault based)**
  
- **Invasive**
  - **First step : depackaging / repackaging**
  - **Optical observation of various layers**
    - » **Localization of critical blocks**
    - » **Reading ROM memory contents**
    - » **Reverse engineering**
  - **Circuit modifications (cutting or adding connections)**
  - **Probing (but number of probes, identification of data, ...)**
  - **Voltage contrast microscopy**
  
- **Semi-invasive**
- **Non-invasive**

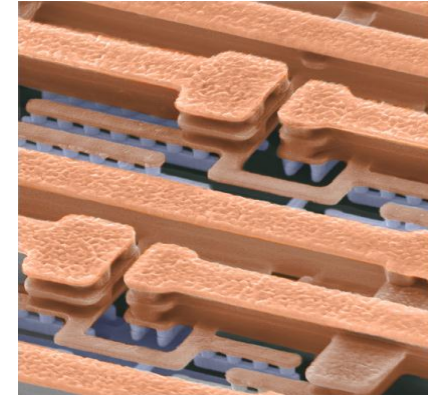
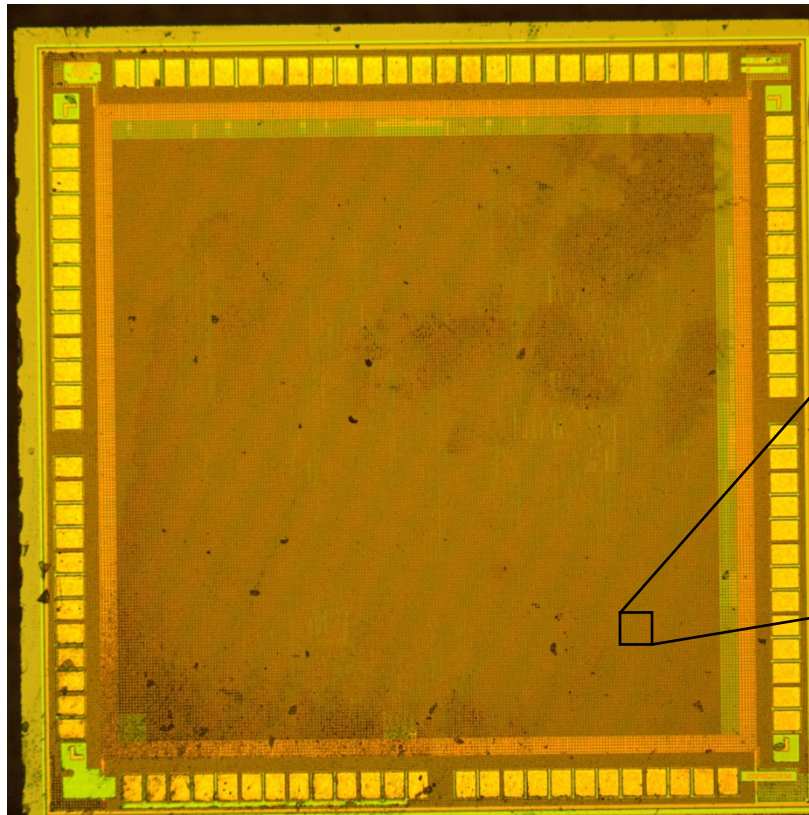
**Sensors (light/UV) can be used to detect intrusion and stop functioning (or even destroy the critical contents)**

**Note: we will not discuss here attacks based on software&networks threats (e.g., viruses, malicious applets)**



# Direct optical observation

---



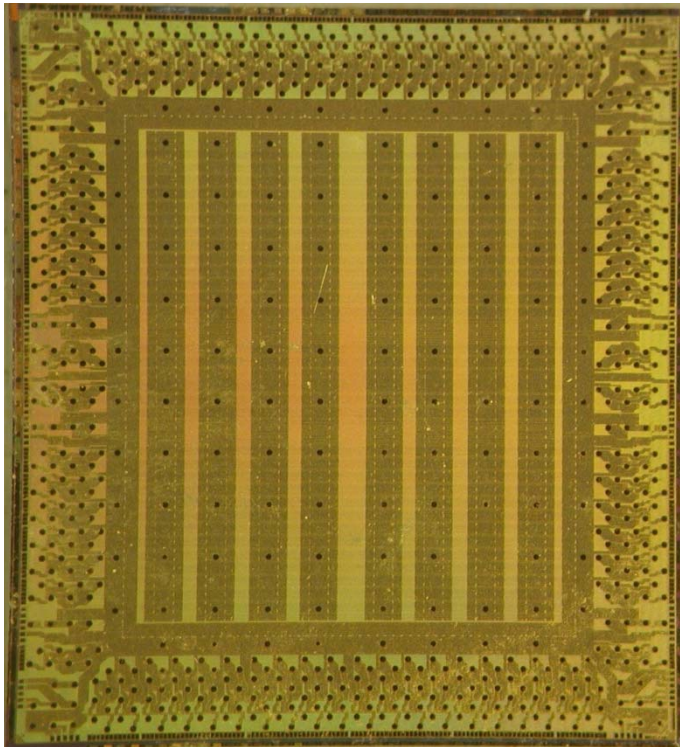
**Dummies: planarization of interconnection levels**

**=> High metal density, reduces optical analysis capabilities of a chip floorplan (localization of critical elements) and accessibility for attacks (e.g. laser)**

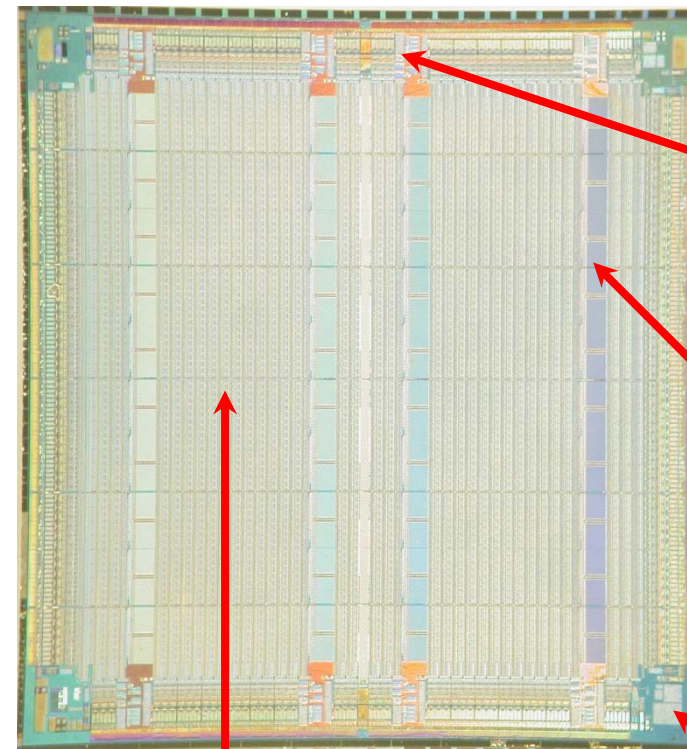
# Layer removal: example on a XC2V1000

---

Before



After - Poly-Silicon layer



**DCMs**

**BRAMs**

**Corner  
logic**

**CLBs**

# Side channels

---

- ❑ **Execution time => Timing Attack (TA)**
- ❑ **Power consumption => Power Attacks (SPA / DPA / CPA)**
- ❑ **Electromagnetic emissions => Electromagnetic Attack (EMA)**
- ❑ **Sound, heat, photons ...**
- ❑ **And many more ?**



# Conditions of side channel exploitation

---

- **Relationship between processed (secret) data and a given physical quantity that can be measured**
  - ◆ It must exist ...
  - ◆ ... and a (good) model of the relationship must be available
  
- **Known executed algorithm, so that predictions can be made**
  - ◆ e.g. AES, RSA ...
  - ◆ Security must no more be based on secret algorithms ...
  
- **Global secret used in independent pieces, so that exhaustive search can be done on separate parts**
  - ◆ Replaces the complexity of a global brute force attack by a local search repeated on the several pieces of the secret

# Side channel analysis

---

## □ Simple Side Channel Analysis

- ◆ Makes use of characteristics that are **directly visible in one measurement trace**.
- ◆ The secret key needs to have some simple, exploitable relationship with the operations that are visible in the measurement trace.

## □ Differential Side Channel Analysis

- ◆ Looks for side channel differences that are not directly visible in one measurement trace => **statistical methods**
- ◆ Targets one specific intermediate result that shows up in a specific part of the measurement traces => selection function.
- ◆ The result of the selection function depends on the known input/output data **and a small number of hypotheses on the secret (key) value**.





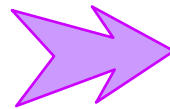
# Timing attacks - principles

---

- **Software level: e.g. If Then Else => execution time may be directly related to the tested condition (e.g. secret key bit !!)**
- **Hardware level: possible similar effect of condition tests in FSMs or data-related computation time in operators**
- **Security requires balanced execution paths and avoiding tests on critical (secret) values**

◆ **Example:**

**If (x=1)  
then a:=a+b;**



**t[0]:=a;  
t[1]:=a+b;  
a:=t[x];**

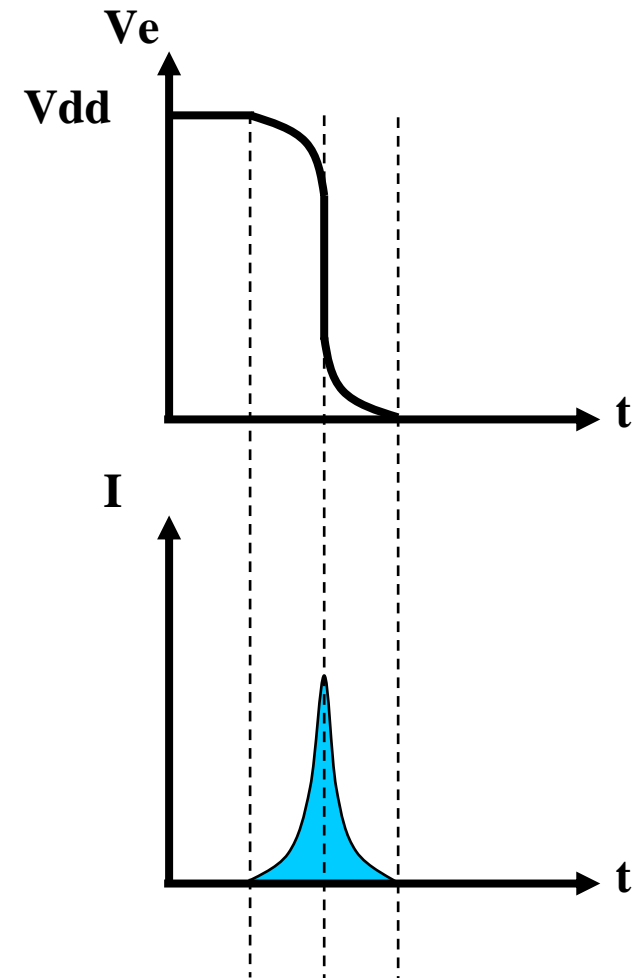
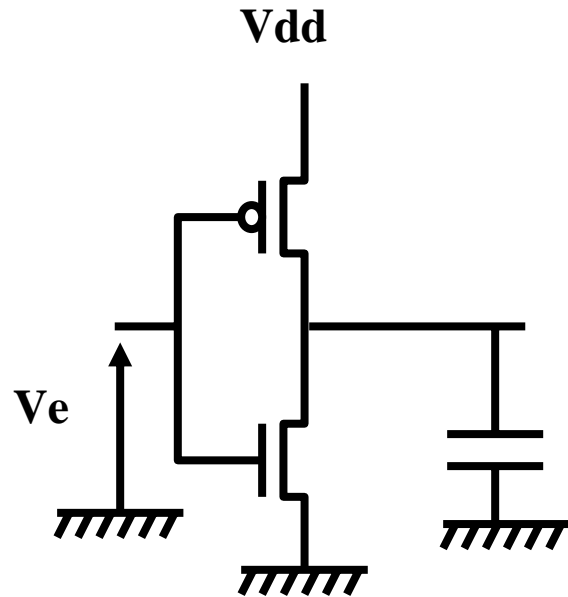
# Architectural features and attacks

---

- **Performance optimizations in processors can help timing attacks**  
e.g., HW optimizations in Pentium 4 were shown to lead to large duration dispersions for constant-time assembly-level implementations
- **Branch predictions (already executed branch has not the same execution time as a new branch)**
- **Cache memories (access to cache has not the same execution time as access to central memory)**
- **Performance counters provide accurate picture**



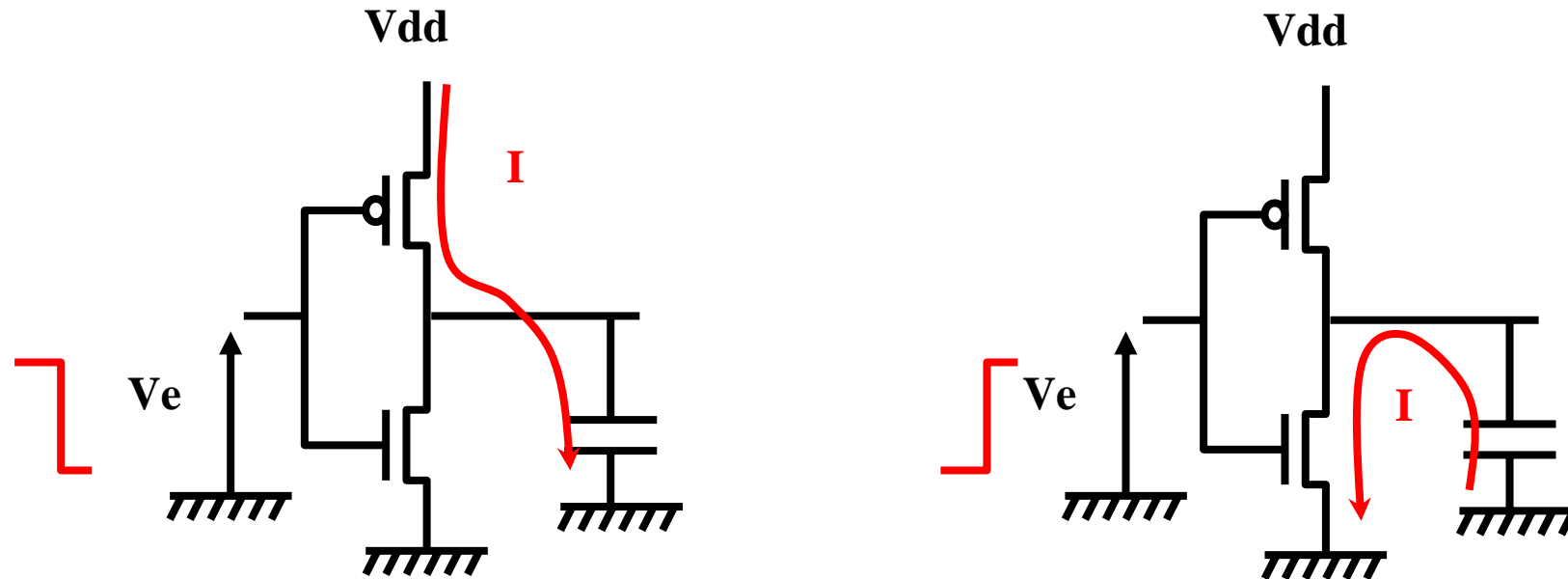
# Power consumption in CMOS - Basics



## □ Three components:

- ◆ Static consumption
- ◆ Dynamic consumption (logic commutations)
  - Short circuit current
  - Load charge/discharge

# Power consumption in CMOS - Dissymmetry

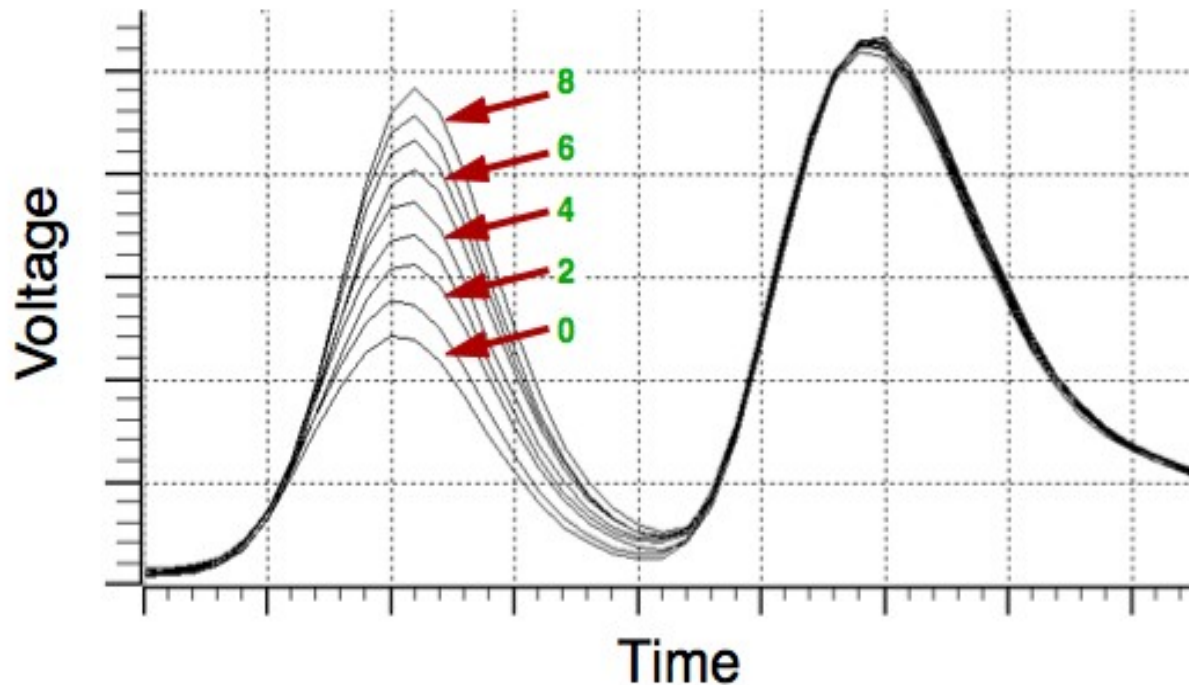


- Opposite logic commutation => different current signature
- Not the same current flow on  $V_{dd}$  and  $V_{ss}$
- Not exactly the same shape (value in time) due to either NMOS or PMOS conduction

# Information leakage: power vs. transitions

- Power consumption curves depend on
  - ◆ Number of simultaneous transitions
  - ◆ Type of transition
  - ◆ + electrical characteristics (node capacitance, ...)

## Hamming Weight or Hamming Distance Leakage



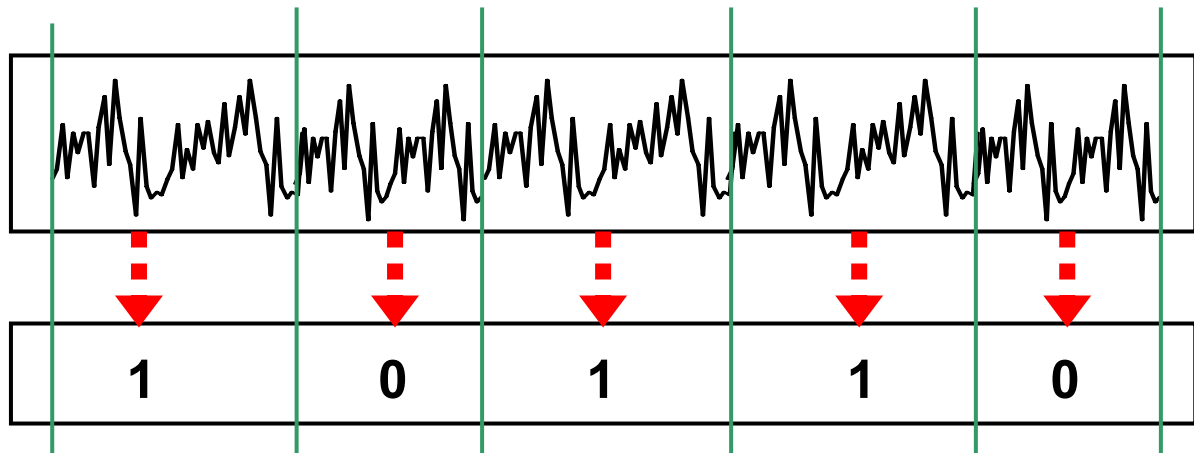
[Kocher]

# SPA (Simple Power Analysis)

---

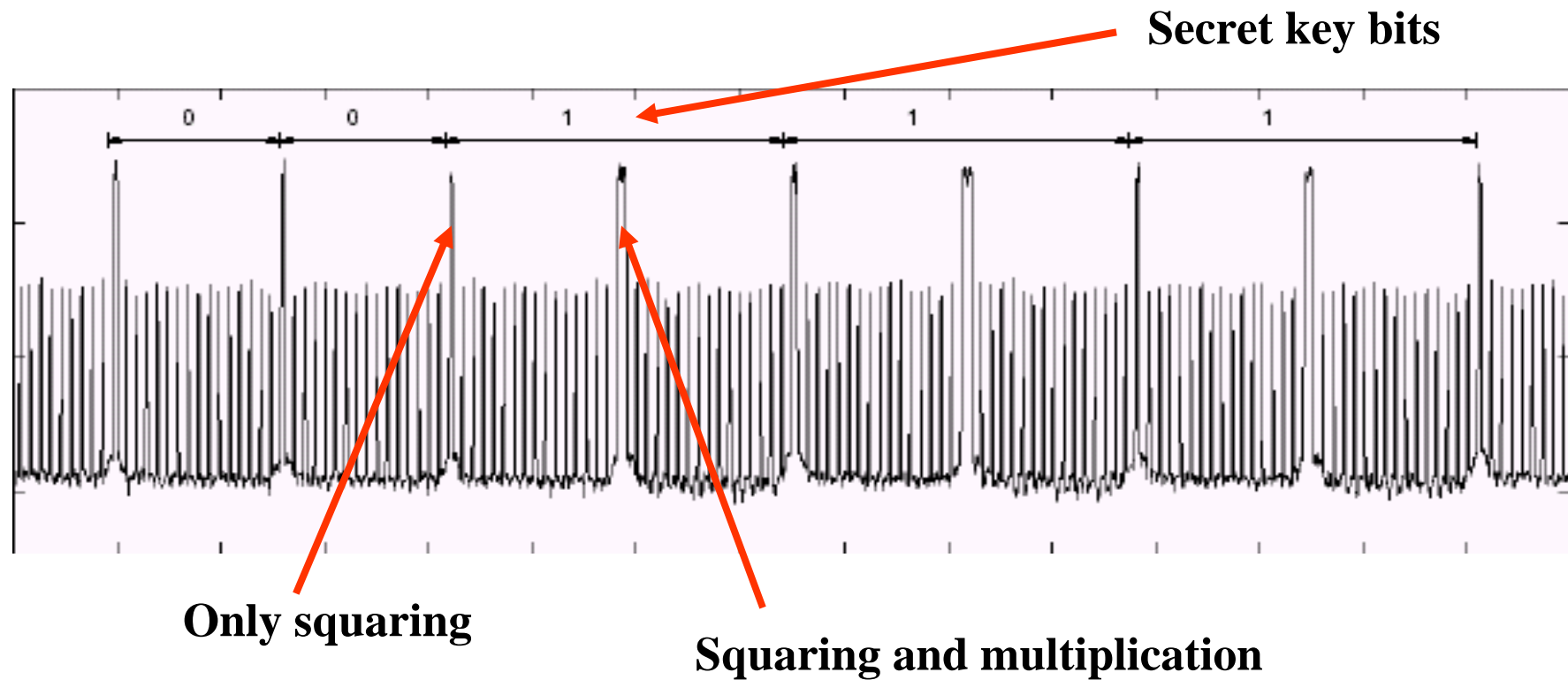
- Monitor the power consumption trace => signatures
- Find translation into
  - ◆ Executed instructions (software)
  - ◆ Manipulated data

Power consumption



# Example of SPA - RSA

## □ RSA computation: supply current trace



# Example of SPA - ECC

## General scalar multiplication algorithm

**Algorithm 1.** Double-and-add algorithm

Input : A point  $P$ , and  $d = (d_{n-1}d_{n-2}\dots d_1d_0)_2$ ,  $d_{n-1} = 1$

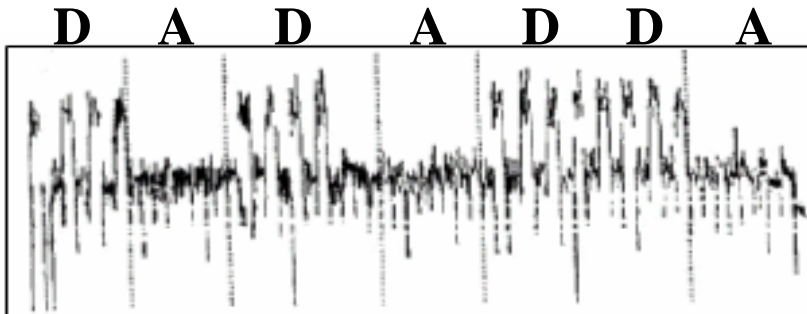
Output :  $dP$

1.  $S = P$
2. For  $i = n - 2$  downto 0
  - 2.1  $S = 2S$
  - 2.2 If  $d_i = 1$ ,  $S = S + P$
3. Return( $S$ )

**d: secret exponent**

**Point Doubling (D):**  
Execution at each round

**Point Addition (A):**  
Execution when bit value is "1"

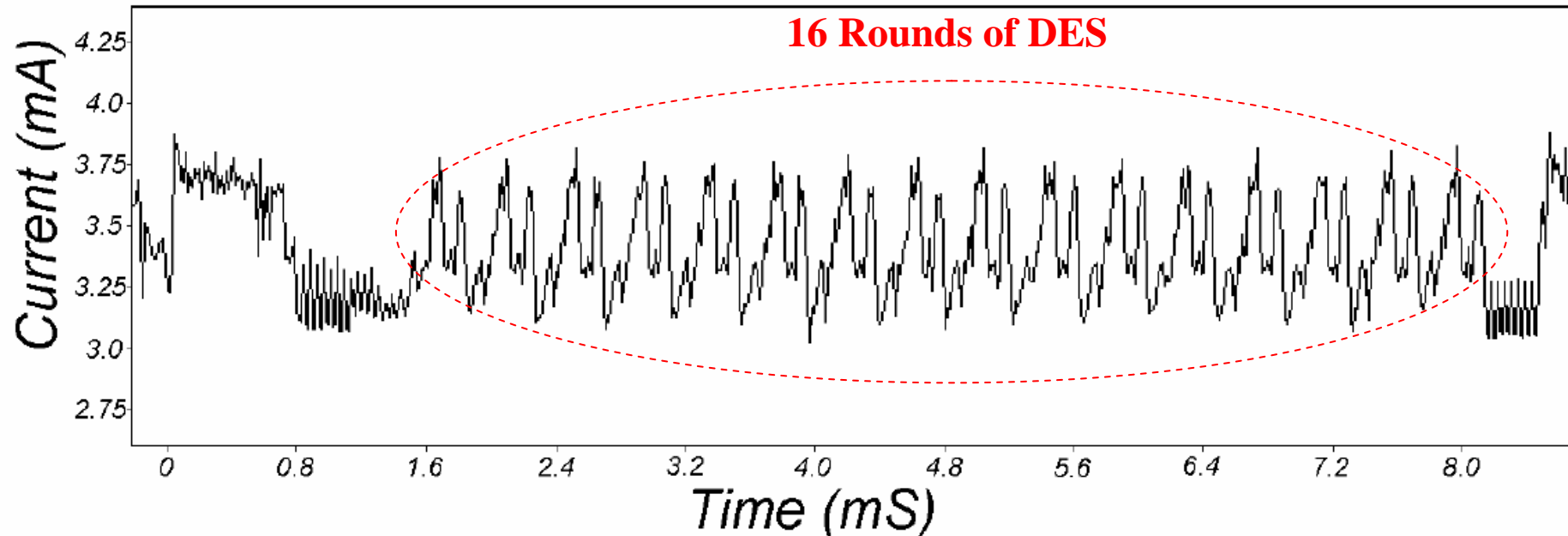


**d = 1101**



# Example of SPA - DES

---



- May also be used to
  - ◆ Synchronize another attack (e.g. laser shot)
  - ◆ Measure the duration of some internal (intermediate) computations (TA)

# DPA (Differential Power Analysis)

---

- **Introduced by P. Kocher (1998)**
- **Data collection phase + data analysis phase**
- **Based on a power consumption model**
- **Attack in pieces (divide and conquer - small parts of the key)**
  - e.g. DES only  $2^6$  choices per Sbox (exhaustive search feasible)



# DPA - Modeling the power consumption

---

## □ Hamming weight model

- ◆ Typically measured on a bus,  $Y=aH(X)+b$   
(Y: power consumption; X: data value; H: Hamming weight)
- ◆ Not very good in CMOS ... but works in practice

## □ Hamming distance model

- ◆  $Y=aH(P\oplus X)+b$  - Accounting for the previous value on the bus (P)
- ◆ Better in CMOS, but requires knowing two successive intermediate values

## □ Other models

- ◆ Hamming weight with fitting coefficients
- ◆ Distinction between rising and falling edges
- ◆ Dedicated models based on the knowledge about a given circuit



# DPA - Basics

---

- **Choice of a power consumption model**
- **DPA can be performed in any algorithm that has operation  $M'=f(M\oplus K)$ , where  $M$ ,  $f$  are known and  $K$  is the segment key**
- **The waveforms are captured by a scope (many executions with many messages) - The selection bit(s) will classify the wave**
  - ◆ e.g., Hypothesis 1: bit is zero / Hypothesis 2: bit is one
  - ◆ A differential trace is calculated for each selection bit
  - ◆ The DPA waveform with the highest peak will validate the hypothesis
- **Typical: 500K+ acquisitions ...**



# Example of DPA - AES

---

- **Number of computations ?**
- **128-bit key (16 bytes) – Each byte used independently as a 8-bit Sbox input (after message XORing)**
- **256 DPA curves to compute per byte,  $16 * 256$  to find the 128-bit key**  
$$2^4 * 2^8 \ll 2^{128} !!$$
- **But**
  - ◆ **Several peaks can be similar (or even a wrong guess may lead to a higher peak)**
    - **Algorithmic noise (implementation model)**
    - **Implementation characteristics**
  - ◆ **Comparing results for several selection bits may help (but do not always agree !)**



# And attacks are more and more acute ...

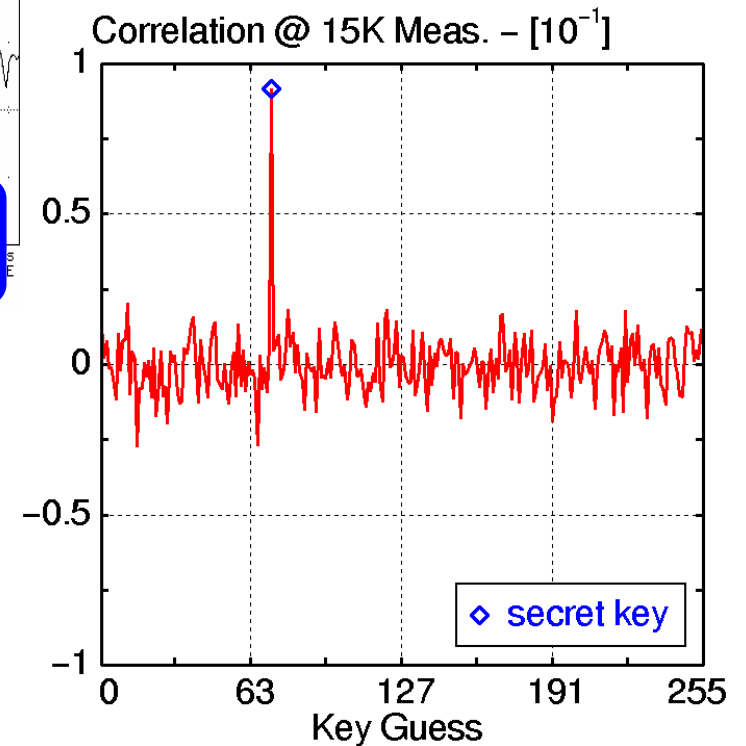
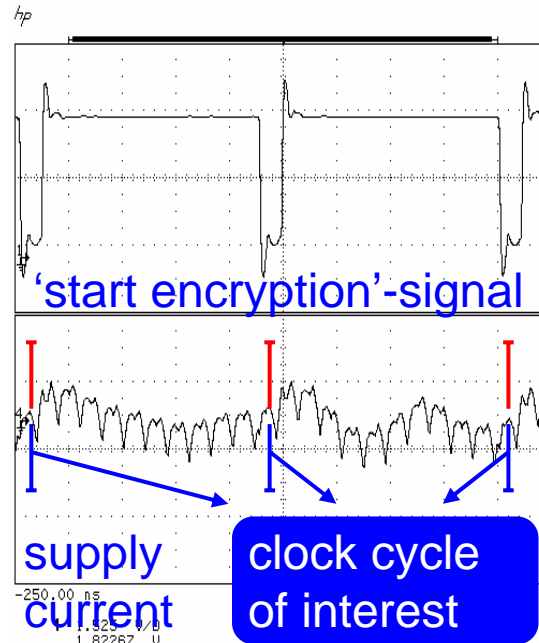
---

- **Mono-bit vs. multi-bit DPA**
  
- **Second order power analysis**
  - ◆ Rather than observing a single consumption time, focus on correlation between two points in time
  - ◆ Can be extended to 3rd, 4th order ...
  
- **CPA: Correlation Power Analysis**
  - ◆ The equation for generating differential waveforms are replaced with correlations (Pearson coefficient instead of mean difference)
  - ◆ Rather than attacking one bit, the attacker tries predicting the Hamming weight of a word
  
- **Template attacks (based on a device characterization)**
  - ◆ A single (a few) sample may be sufficient (suited to stream ciphers, and cases with key re-use limited by system level protocols)



# Power analysis - AES

- Unprotected ASIC AES with 128-bit datapath, key scheduling
- Measurement: I<sub>peak</sub> in round 11
- Estimation: HamDistance of 8 internal bits
- Comparison: correlation
- Key bits easily found despite algorithmic noise
- 128-bit key under 3 min.



Source: K. Tiri, Intel



# Electromagnetic attacks (SEMA – DEMA)

---



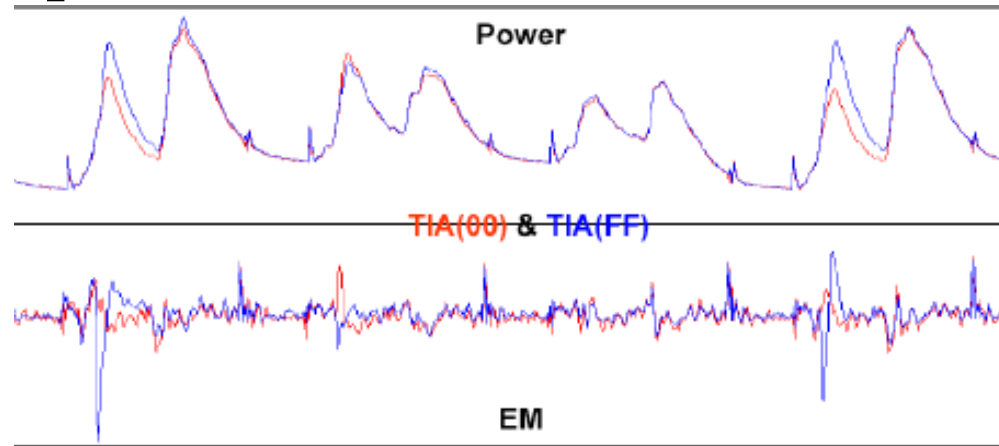
**Same principles as power attacks, with a different measurement equipment giving more precise (more localized) internal measures and bypassing current smoothers ... but experimentally more difficult to put in practice.**



# Comparison power/EM

---

- EM more local (depending on setup and probe), with wider bandwidth (~100 MHz power, ~1 GHz EM) => more data in the recorded signal



- No direct connection required for EM (e.g. better for FPGA PCB !)
- Magnetic field better for smartcard analysis than Electric field, but small amplitude and SNR => low noise amplification required + trade-off between precision (probe size) and signal amplitude
- Many parameters: probe size and orientation, measurement position on chip, distance to chip, time interval for analysis, ...

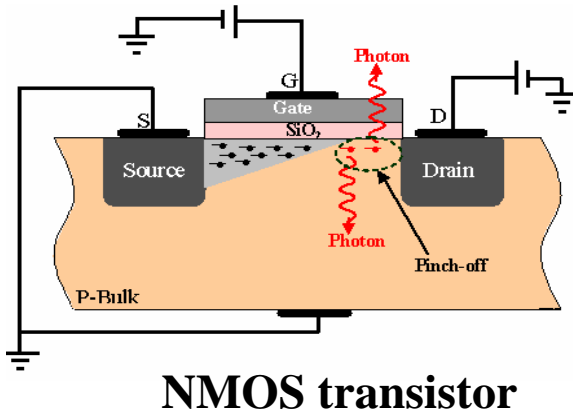
# EMA "success story" - example

---

- **EAL4 product - AVA\_VAN.5 robustness level**
- **Protected ECC crypto-processor with "add always" implementation or RSA with "multiply always"**
- **About 2 weeks of (manual/visual) EM measurement analysis by expert**
- **Broken key – due to difference in storage address (least significant bit) for the useful and useless additions or multiplications**



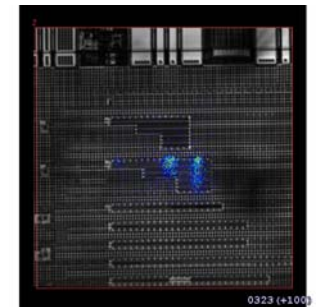
# Attacks based on light emission (photons)



Failure analysis equipments  
Hamamatsu – Tri PHEMOS



- Spatial analysis of photon accumulation
- Dynamic analysis techniques (TRE, PICA)
  - ◆ Signal propagation analysis
  - ◆ Functional analysis (function behavior - reverse engineering)

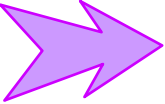


Courtesy J. Di-Battista (PhD defense - April 2011)

# Perturbation-based attacks

---

## □ Modifying the nominal working conditions

- ◆ Temperature
  - ◆ Power voltage
  - ◆ Clock frequency (overclocking) or clock shape
  - ◆ ...
-  Indirect fault/error induction (e.g. timing errors)

Sensors (temperature/voltage) can be used to detect abnormal conditions

## □ Directly inducing faults/errors

- ◆ External electrical perturbations – glitches (power, clock, ...)  
Non intrusive but sensors may be used on some critical lines  
+ filtering (e.g. power line)
- ◆ Optical perturbations – (focused) white flash light, laser, UV, X Ray
- ◆ Electromagnetic sources, particles
- ◆ ...

# Ideal fault induction technique ...

---

- **Location control (x,y)**
- **Timing control (start, duration)**
- **Fault type control (stuck at, bit flip, etc.)**
- **Focalization control (number of faulty bits)**
- **Reproducibility**
- **Low cost**
- **Easy to develop and use**

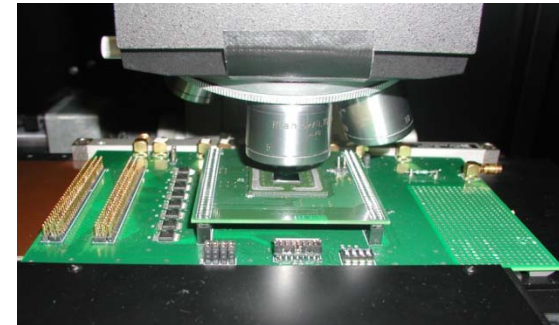


# Laser-based attack: preparation

---

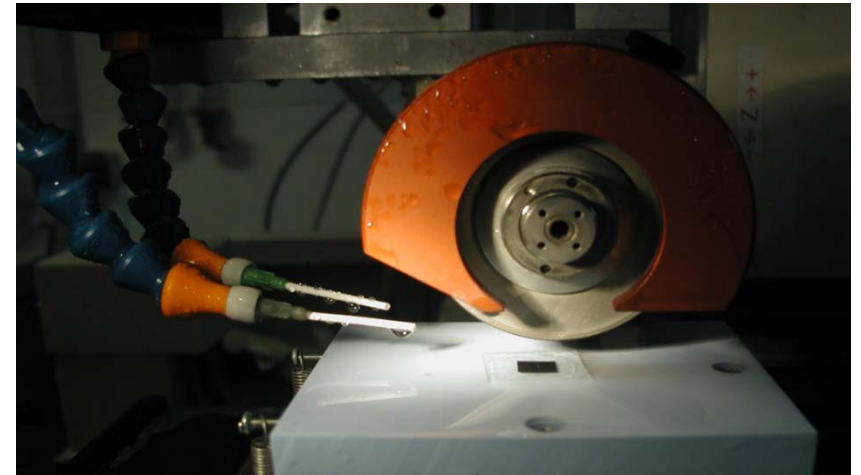
## □ Example on a Virtex chip (just unpackaged)

- ◆ Substrate @ 790  $\mu\text{m}$
- ◆  $\lambda \approx 900 \text{ nm}$
- ◆  $P \approx \text{some Watts}$
- ◆  $\varnothing = 40 \mu\text{m}$  and  $8 \mu\text{m}$
- ◆ Backside attack  $\Rightarrow$  No error



## □ Substrate thinning

- ◆ Die thinned by a mechanical process
- ◆ Residual thickness of  $30 \mu\text{m} \pm 1 \mu\text{m}$
- ◆ Successful backside attacks



# Fault-based attacks: basics

---

- **In general a combination of perturbation and cryptanalysis**  
**=> fault creation + (potential) exploitation**
  
- **Example of the Bellcore attack on CRT-based RSA**
  - ◆ **One correct encryption result**
  - ◆ **One erroneous result from the same encryption (using e.g. a laser)**
  - ◆ **A simple GCD computation ... and the 1024-bit key is no more secret !**
  - ◆ **Few constraints on the injection: hit only one of the exponentiations**
  
- **Predictive robustness analysis: requires a good knowledge of injected faults**
  
- **FA (erroneous result directly exploited) – DFA (erroneous vs. correct result)**

# When is the attack successful ?

---

- When the fault is injected and the computation is corrupted ?

**Not yet !**

- When the erroneous result is obtained ?

**Yes ! IF exploitable ...**

- So the erroneous result must

- ◆ Either be hidden (but may give useful indications to the hacker)
- ◆ Or be corrected
- ◆ ... Or the hacker may be misled (erroneous result replaced by another that cannot be exploited)
- ◆ Note: delays induced by detection + recovery => information to the hacker





# Use of induced errors: DFA, but not only !

---

- Round reduction, RNG output forcing => easy cryptanalysis
- PIN counter corruption => unlimited trials
- Hardware or software protection bypass ...
- Behavior analysis ("safe-error" attack): if the attack effect is controlled, e.g. stuck-at-0 bit, a normal or abnormal behavior indicates detection or not of the attack, thus the initial value of the attacked bit (**even without output result to analyze**).
- Direct hacking: counter modifications (e.g. money in e-purse)



---

# Modeling / Characterizing errors



# Permanent vs. transient faults/errors

---

- Both can be a security threat (need for an efficient test !)
  
- Attacks can be based on both (physical modifications vs. bit modifications)
  
- Logical modifications often more dangerous => focus of the sequel
  - ◆ Many different modifications can be induced on a **single** circuit sample
  - ◆ Possibility of efficient time-limited perturbations (see for example Bellcore attack on RSA – a permanent modification would disturb both parts of the computation)
  
- Impact of the technology: permanent/transient or remanent



# Induced errors – logic point of view

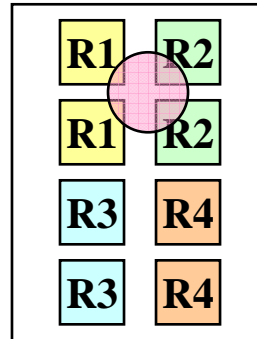
---

- **Errors do not really depend on the physical perturbation technique => soft errors when non invasive**
  - ◆ **Direct induction in memory elements: SEUs/MBUs**
  - ◆ **Induction in combinatorial logic: SET then propagation**
  - ◆ **Final effect: erroneous bit(s) in register(s)**
  
- **Multiplicity depends on the source and fault location**
- **Error types depend on the component => SRAM-based FPGAs**
  
- **Error models assumed in published attack schemes are not always realistic (w.r.t. current fault induction techniques) – e.g. one single particular bit forced at a given value at a given cycle during the computation ... or only one bit changed at a time ...**

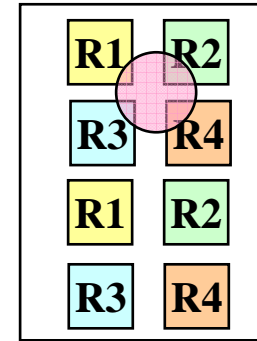
# Laser attack: SEU or multiple bit-flips ??

## □ Depends on

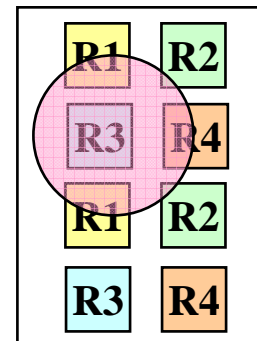
- ◆ Laser focus
- ◆ Placement/routing
- ◆ Cell sensitivity
- ◆ ...



P&R-1, focus 1  
=> mult. up to  
2 per element,  
2 elements



P&R-2, focus 1  
=> mult. up to  
1 per element,  
4 elements



P&R-2, focus 2  
=> mult. up to  
2 per element,  
4 elements

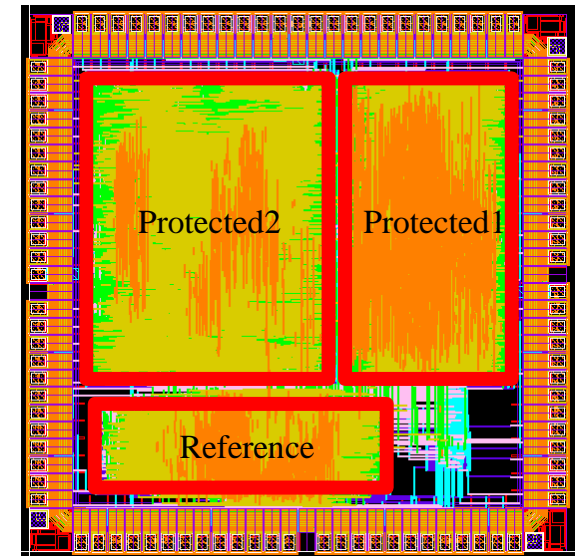
## □ High-level design: no information on P&R

=> assumptions / limitations (e.g. limited to the elements in a given register), but gives constraints on P&R for coherence (and may be optimistic w.r.t. actual attacks with large focus)

# Case study 1: custom implementation

---

- **Montgomery multiplier (n = 512) – 130 nm ST technology**
- **3 versions**
  - ◆ **One reference version without protections**
  - ◆ **2 versions protected by parity-based encodings**
- **Faults induced by laser**
- **Main goals:**
  - ◆ **Analyze the practical effectiveness of such a protection scheme**
  - ◆ **Analyze the type of faults/errors actually induced in the device (no data available in the literature for complex digital circuits)**



R. Leveugle, A. Ammari, V. Maingot, E. Teyssou, P. Moitrel, C. Mourtel, N. Feyt, J.-B. Rigaud, A. Tria  
"Experimental evaluation of protections against laser-induced faults and consequences on fault modelling"  
Design, Automation and Test in Europe Conference (DATE), April 16-20, 2007, pp. 1587-1592

# Implementation characteristics

---

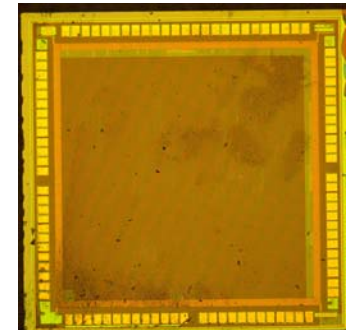
Circuit	Area (mm <sup>2</sup> )	# cells	# equivalent gates
Reference	0.434	26,589	55,751
Protected 1	0.827	46,084	116,796
Protected 2	1.282	59,600	160,849

- Overheads mainly due to the parity prediction in the large systolic array
- Protection limited to odd number of erroneous bits (due either to bit-flips, or erroneous signal propagation and latching)
- Detection if this condition occurs in at least one block (potentially several blocks)

# Campaign specification

---

- **No area or time a priori more critical than another from the application point of view => global scan of the blocks during the whole computation process**
  - => black box approach**
  
- **Limited time available**
  - ◆ **Medium focus ( $147\mu\text{m} \times 145\mu\text{m}$ ) => 27, 45 and 63 zones for resp. Reference, Protected1 and Protected2 versions**
  - ◆ **Only 10 cycles uniformly selected during the computation**
  - ◆ **Same input data (A, B, N) for all experiments**
  
- **5 shots per spatial and temporal position => 1350, 2250 or 3150 shots per block**
  
- **Energy level at "zero" => "leakage beam"... (in spite of front end shot on dummies, 6 metal layers )**





# Classification results

Circuit	Undetected wrong results	No answer	Detected wrong results	No effect
Reference	89.58 %	10.42 %	--	0 %
Protected 1	0.27 %	0 %	99.73 %	0 %
Protected 2	0.32 %	0 %	99.68 %	0 %

## □ Protected 1

- ◆ 6 wrong results without alarm
- ◆ Same zone, at five different cycles distributed from the beginning until almost the end of the multiplication execution

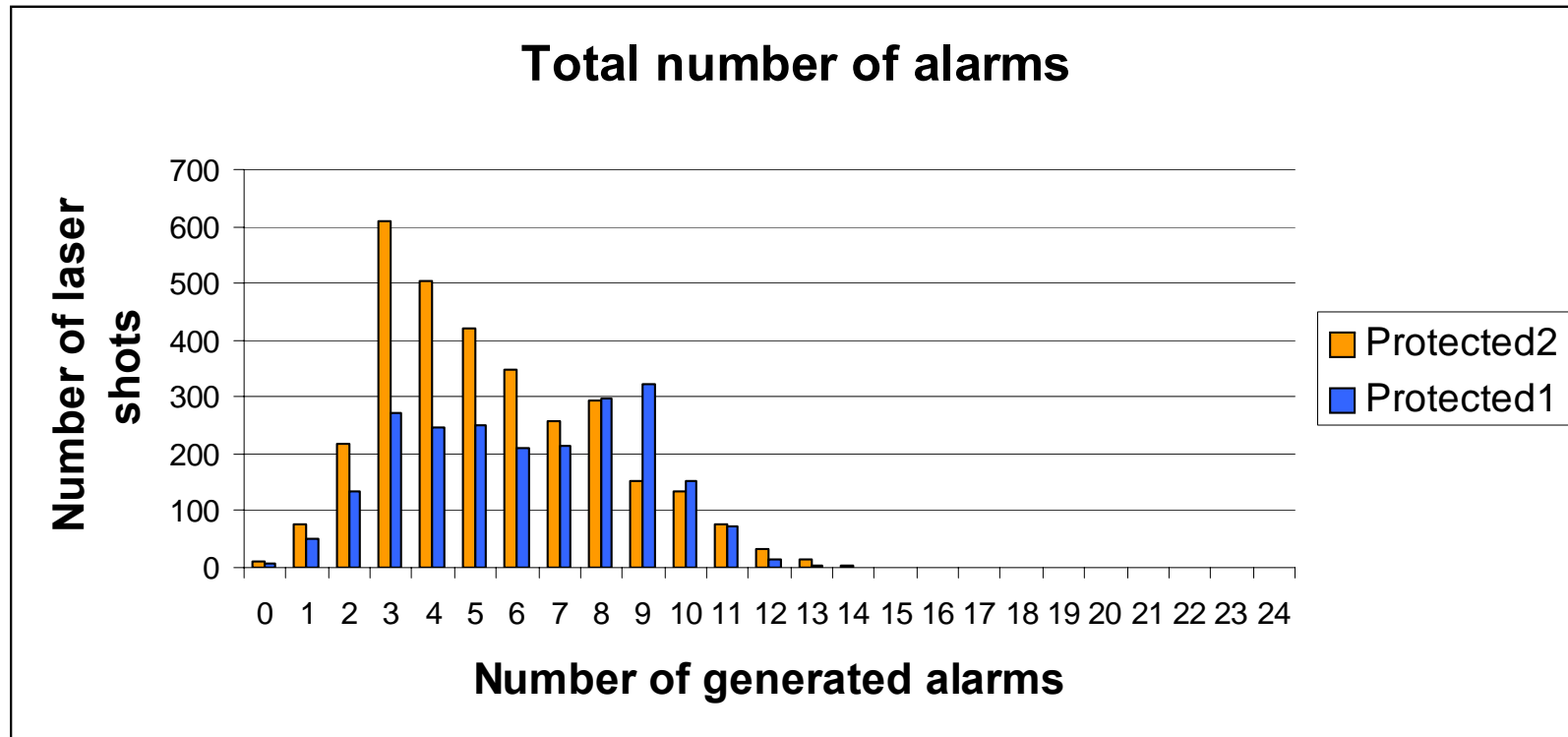
## □ Protected 2

- ◆ Same conclusions: 10 wrong results, on two zones at seven different cycles

## □ Multiple-bit errors with even parity are quite easy to generate ...



# Repartition w.r.t. number of asserted alarms

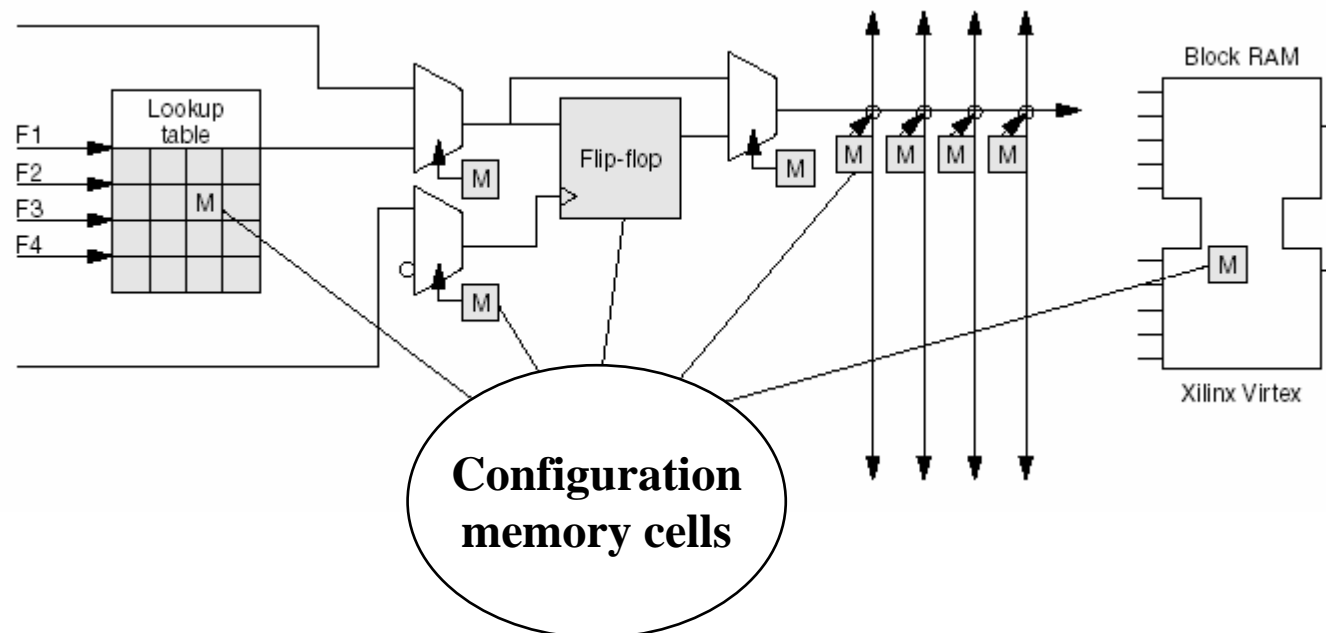


- ❑ Multiple errors are the general case
- ❑ High level of multiplicity

# Other case-study: SRAM-based FPGAs

## □ Faults can modify:

- ◆ In ASICs: mainly processed data (or control states).
- ◆ In SRAM-based FPGAs: both processed data and function definition (configuration errors)



# Configuration errors in SRAM-based FPGAs

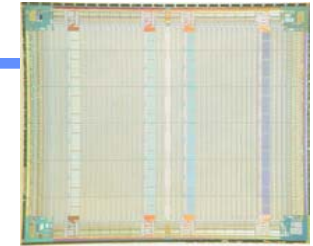
---

- Configuration errors do not necessarily lead to application errors (unused FPGA resources, implicit redundancy ...)
  
- They may also lead to several types of consequences
  - ◆ Design modifications (not only data disturbance)
  - ◆ SEFIs (Single Event Functional Interrupts)
    - Non persistent (correcting the configuration is sufficient to recover, e.g. combinatorial part modification)
    - Persistent (a global restart is necessary after correcting the configuration, e.g. modification of a FSM state)
  
- Focus in the following on configuration errors in CLBs (logic functions – computation and interconnections)



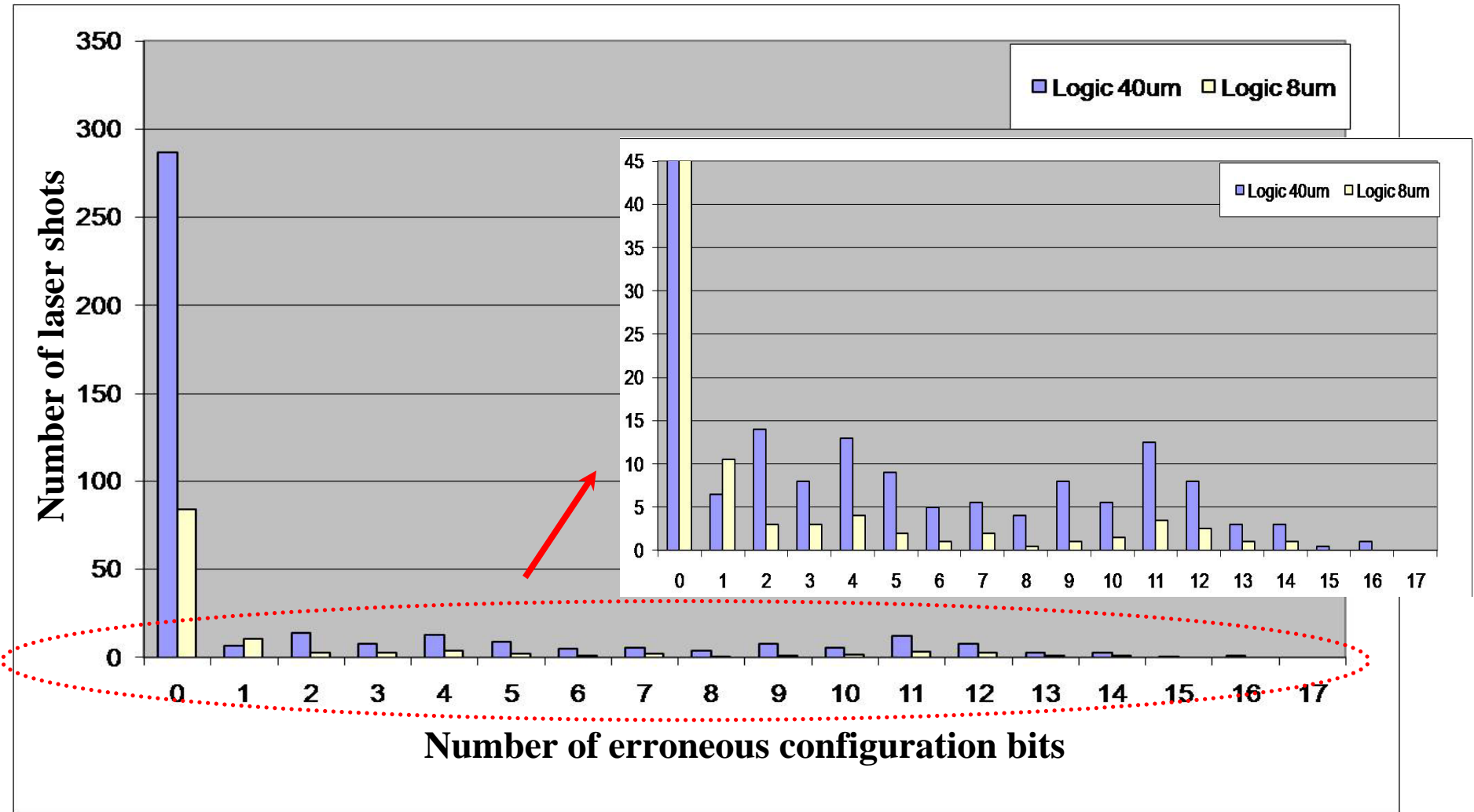
# Case study 2

---

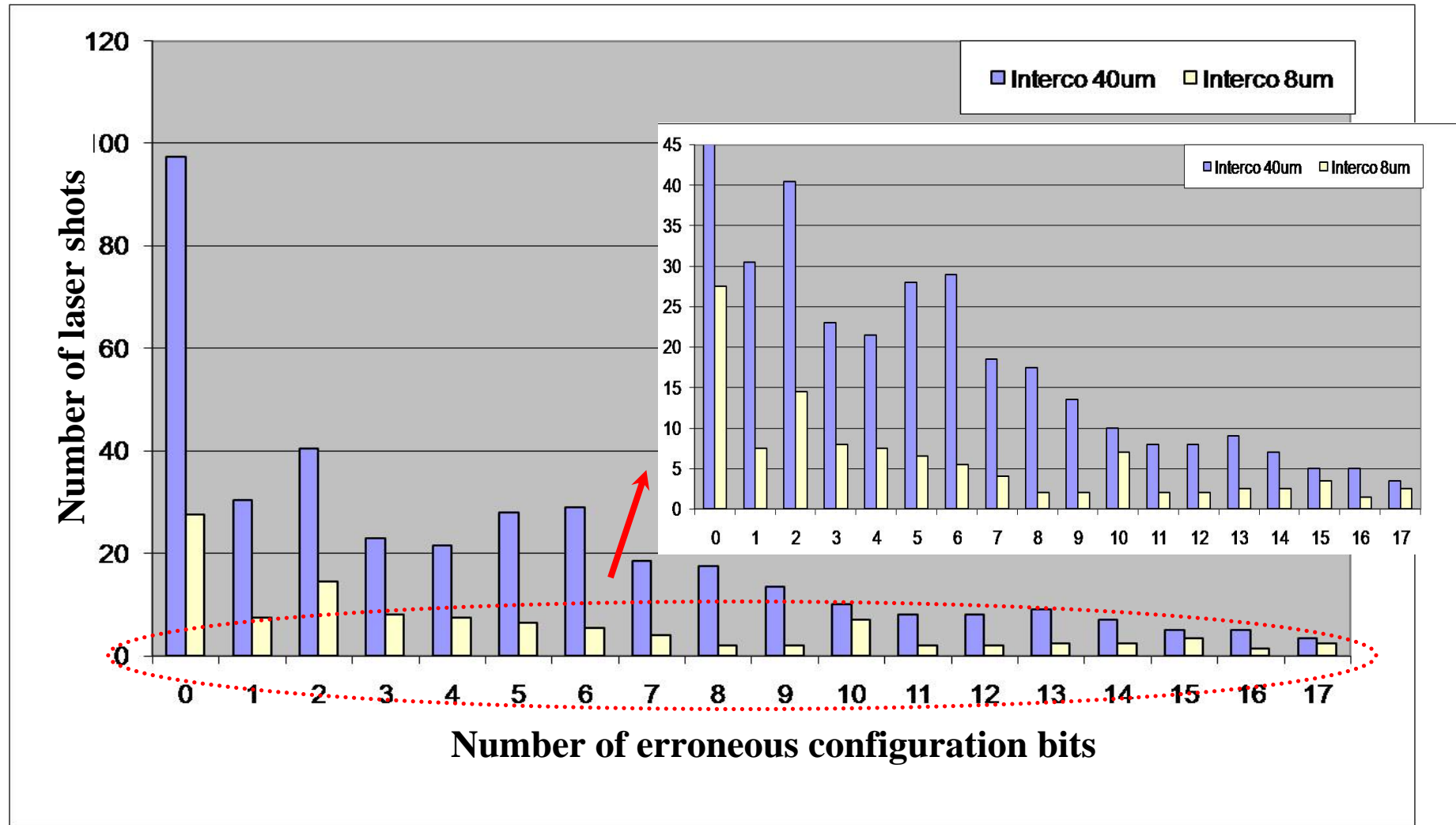


- ❑ **Laser shots performed on a Virtex II XC2V1000 (150 nm CMOS 8-layer metal process)**
- ❑ **Similar conclusions with several lasers**
- ❑ **Most sensitive sub-blocks: CLB (logic + interco.), BRAM**
- ❑ **Characterization of patterns obtained after single shots**

# Configuration errors in logic (single shots)



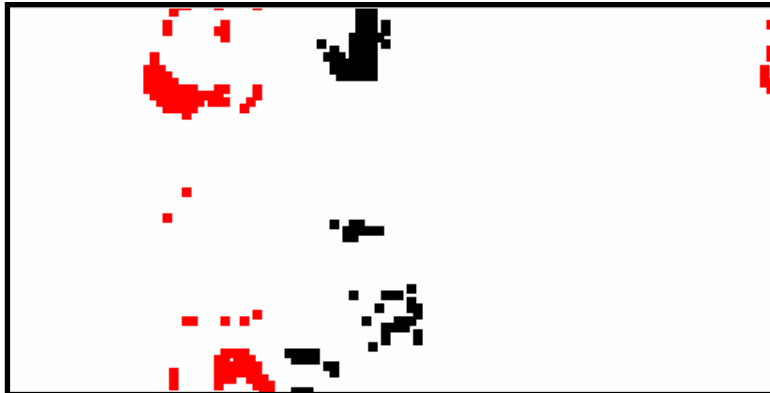
# Configuration errors in interco. (single shots)



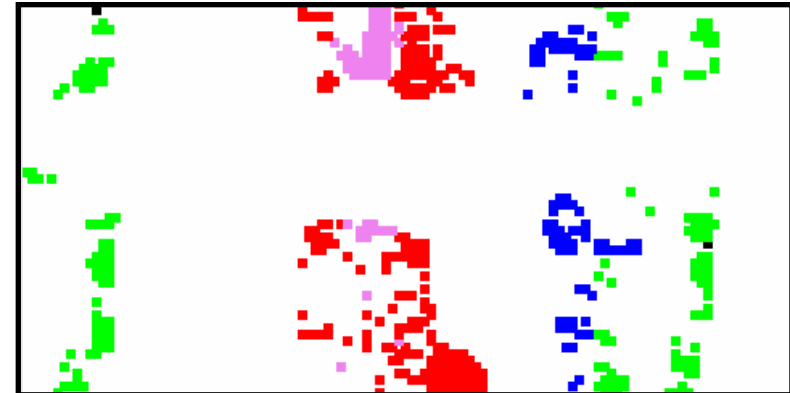
# Special case: single bit flips

---

- **High probability of multiple bit flips ... but possibility to flip only one bit per shot**



**Single bit flips in logic parts**



**Single bit flips in interconnections**

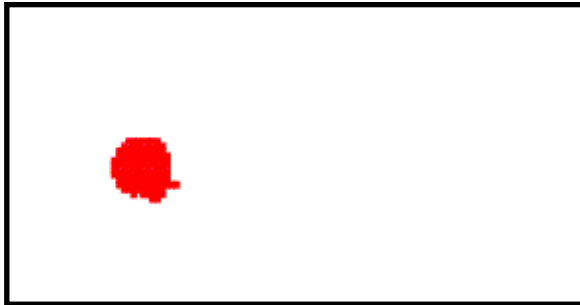
- **Number of positions leading to this special case:**
  - ◆ 10% of the total number of laser shots
  - ◆ 25% of the total number of modified configurations



# Effective sensitive area - Shape

---

- The sensitive area shape depends on the bit value



**Bit initially at 1**



**Bit initially at 0**

..... and the notion of diameter has to be revisited

- The same shape was observed no matter the bit functionality in CLBs and also in BRAMs

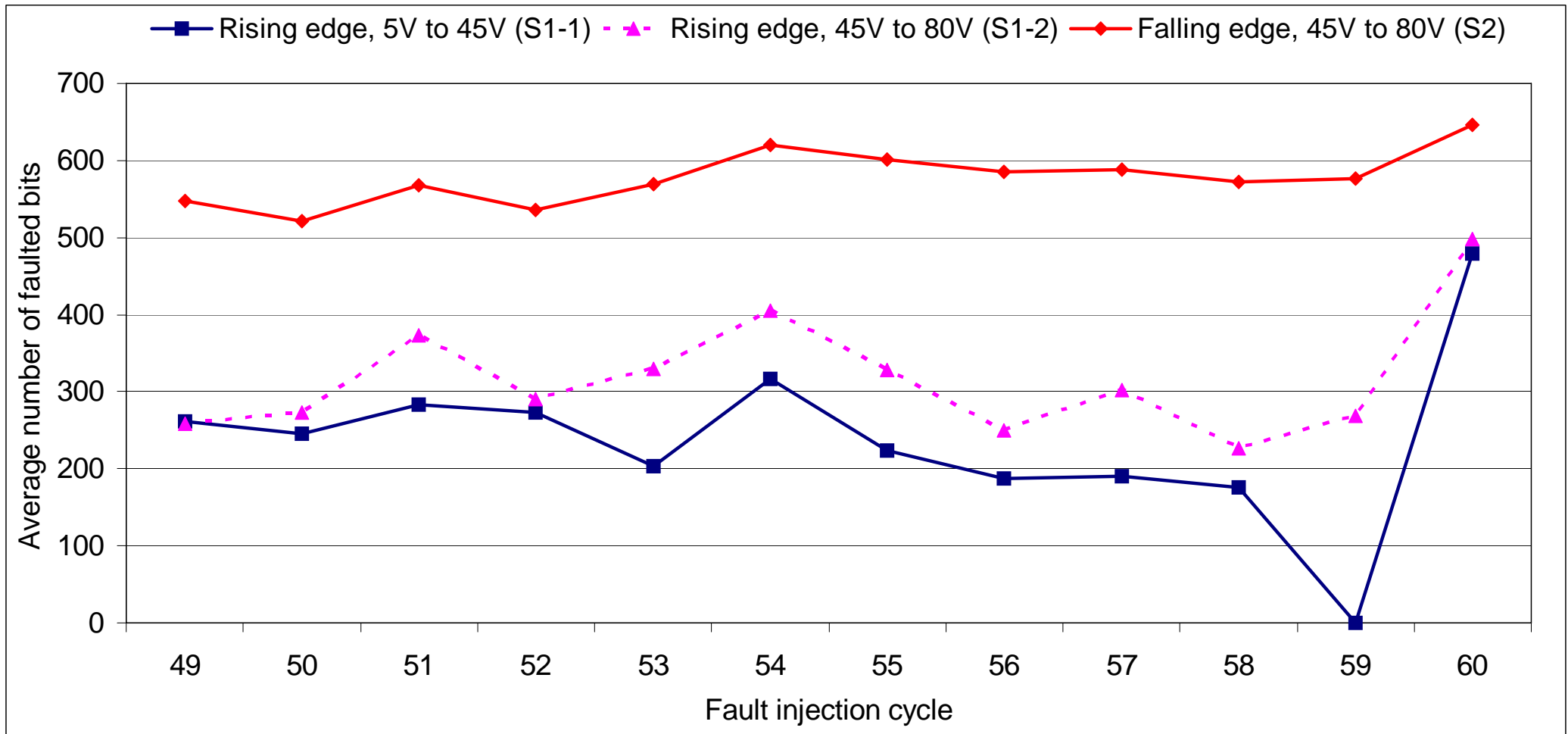
# Laser vs. Power Glitch attacks

---

- **All types of perturbations do not lead to the same effects (and thus to the same error model)**
- **Laser attack: multiple bit errors in the configuration  
=> remanent faults, to be taken care of during counter-measure implementations (e.g. time redundancy)**
- **Glitch attacks (overshoots) on a Virtex II: no configuration error, only bit flips in the user layer  
=> only transient faults to be considered  
=> little spatial control, very high multiplicity  
=> effects depend on the overshoot amplitude, but also on the attack synchronization**



# Power glitch attacks on a AES coprocessor



---

# Additional attacks



# A special case: reconfigurable hardware

---

- **In case of SRAM-based or Flash-based FPGA**
- **Specific possible attack by "downgrade", i.e. replacing the FPGA configuration by an older one (correctly encrypted and with correct integrity checks) => may maintain functionality but remove security "patches" correcting previously identified security holes**
- **Need for accurate versioning !!**

# Test-related problems

---

- **Independent internal clock generation (to make synchronized attacks more difficult) vs. need for external clocks during test**
  
- **Boundary scan and internal scan access make test easier, but give access to all internal data**
  - ◆ **Can be used through external ports**
  - ◆ **Can be activated with only a few probes in an invasive way (serial approach)**



# Securing test modes ...

---

- **Secure test mode activation (secret key, ...) rather than fuses**
- **Detection of unexpected shifts (due to probing)**
- **(Variable, random) scrambling of scan chains**
- **Dummy bits in scan chain (detection of invalid shifted patterns)**
- ...
- **Avoid scan techniques when possible: self-test (BIST, SBST)**



# Design constraints: link with security ??

---

- Area and yield ? **Yes, limits protections**
- Speed (clock frequency / computing power) ? **Yes, limits protections and/or increases susceptibility to faults**
- Power/energy consumption and heat dissipation ?
  - ◆ Dynamic consumption: **Yes, side channel**
  - ◆ Temperature: **Partially, temperature-induced faults**
- Pin number (off-chip interconnections) ? Not directly
- On-chip interconnections ? **Yes, in particular dummies**
- Testability ? **Yes, potential backdoor**
- Electromagnetic characteristics (emission, susceptibility) ?  
**Yes, side channel and possible perturbation**



---

# **Protections (counter-measures) against several types of attacks**



# Secure circuits

---

- **Secure systems are nowadays directly in the hand of the (potentially malicious) user => counter-measures must be built into the system itself (no bunker!)**
  
- **"Classical" protection styles (but adapted), especially against faults/errors**
  
- **Specific protections (active)**
  - ◆ **Sensors (voltage levels, glitch, light, temperature, ...)**
  - ◆ **On-chip encryption of data**
  - ◆ **...**
  
- **Specific protections (design methodology)**
  - ◆ **Internal clock generation (problem with testing requirements !)**
  - ◆ **Restrictions in design styles (dynamic logic, ...)**
  - ◆ **Shielding, Memory scrambling, ...**

**Multi-level:**  
- **Software**  
- **OS**  
- **Hardware**

# Limitations of counter-measures

---

- **Never 100% protection/security – just make the attacks harder**
  
- **Still an empirical selection process**
  - ◆ **No global optimization of overheads**
  - ◆ **A protection against a given attack can ease another one (e.g. error detecting codes w.r.t. power-based attacks)**
  
- **All types of possible attacks must be simultaneously addressed (the hacker will use the easiest way to success)**

# Counter-measures against SCA

---

- **Attacker always needs an initial guess to compare his measurements to - Protection: make measurements unguessable or constant (data independent)**
- **Several levels**
  - ◆ **Algorithmic level**
  - ◆ **Architecture level**
  - ◆ **Gate level**
  - ◆ **...**
- **Examples**
  - ◆ **Avoid conditional execution**
  - ◆ **Use constant-time (i.e. worst-case time) programs**
  - ◆ **Use dedicated “non-leaking” logic**
  - ◆ **Randomize execution flow (SW or HW) – But resynchronization attacks are possible**
  - ◆ **Add noise... but this will only require more samples for a successful attack**



# Protections against power analysis

---

## □ Hardware:

- ◆ **Main idea: balanced switching (systematic simultaneous 1 -> 0 and 0 -> 1 transitions on SAME load ...)**

**Not so easy ! Specific design techniques/tools,  
load balancing (P&R ...), ... but increasing process variations !  
+ must avoid evaluation anticipation (internal Z nodes), ...**

## ◆ Other ideas:

- **Reduced synchronization by variable/random jitter (not just noise !!)**
- **Masking of actual algorithmic values (manipulated data are not directly correlated to the key value) – but still vulnerable to DPA**

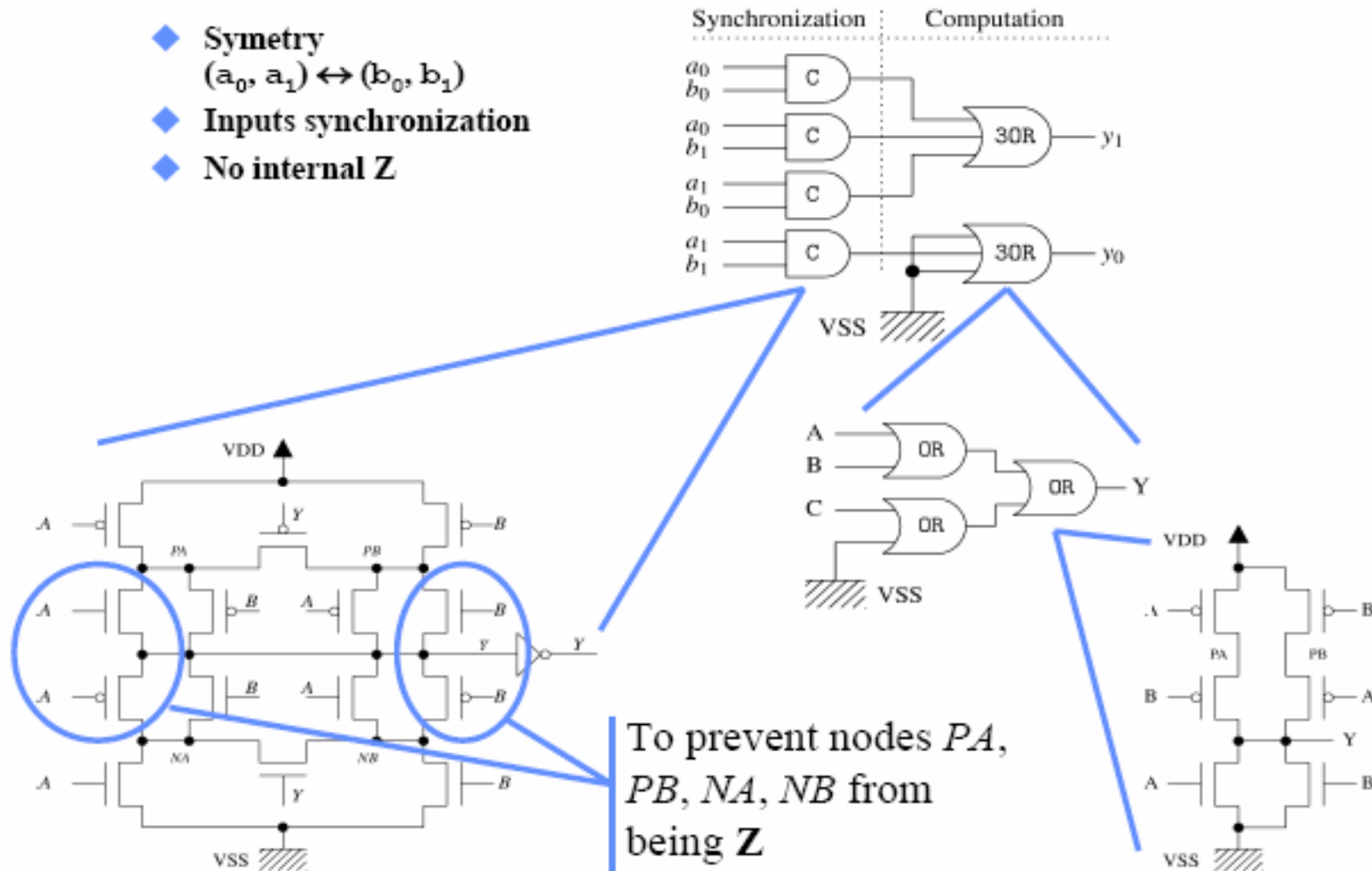
## □ Software:

- ◆ **Specific coding techniques**
- ◆ **Not just compiling (assembling level interventions)**



# Example of secured NAND gate

- ◆ Symetry  
( $a_0, a_1$ )  $\leftrightarrow$  ( $b_0, b_1$ )
- ◆ Inputs synchronization
- ◆ No internal Z



From ECoFac 2006 presentation – S. Guilley et al. (GET/ENST)

# Towards a dependable system (w.r.t. errors)

---

- **Many approaches/techniques, a few major categories**

- ◆ **Spatial redundancy, replication (massive redundancy)**
- ◆ **Information redundancy (coding)**
- ◆ **Timing redundancy**
- +
- ◆ **Assertion-based on-line checking**
- ◆ **Control-flow checking**
- ◆ **etc. ...**

**Hardware  
and/or  
software**

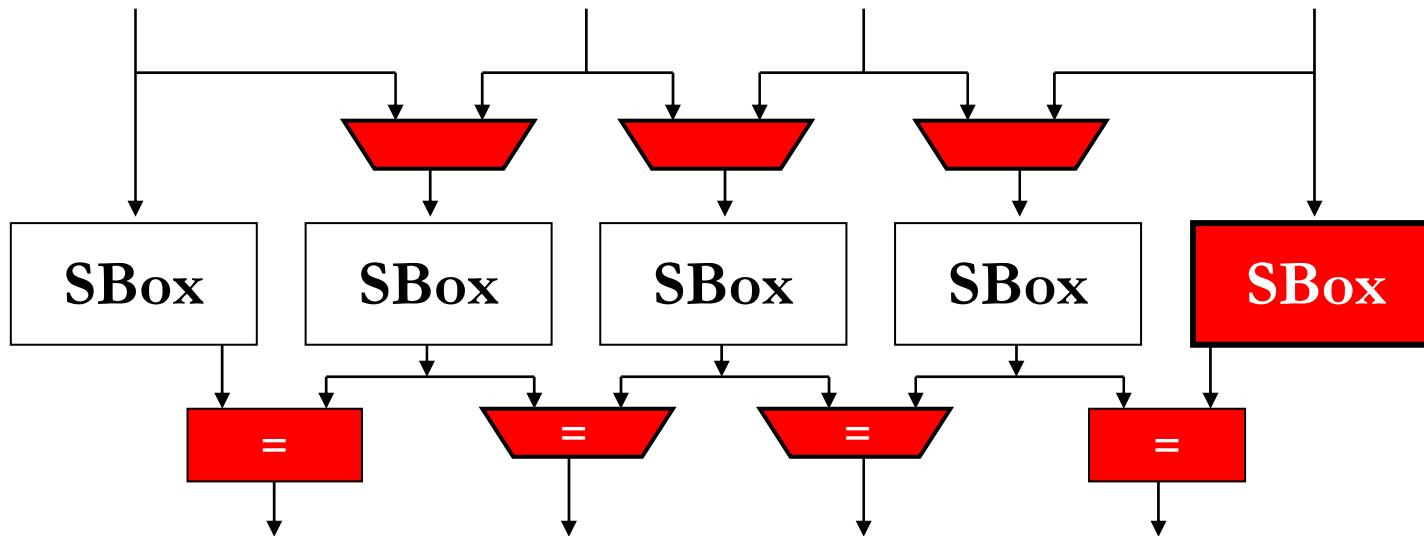
- **Choice depends on application constraints (overheads, ...) and on dependability specification**

- ◆ **Detection only**
- ◆ **Tolerance by detection + recovery (retry, checkpointing, ...)**
- ◆ **Tolerance by masking (voting) ... taking care of fault accumulation ...**



# Partial redundancy example

---



- **Dynamic comparison with additional component**
- ☺ **Limited overhead and good detection capability of permanent faults**
- ☹ **Limited detection capability of transient faults**

[Di Natale, Flottes, Rouzeyre. WDSN 2007]

Courtesy P. Maistri





# Using error correcting/detecting codes

---

- **One of the typical methods of protection**
  - ◆ **Well established theory – many codes available, protection efficiency against errors is theoretically well known**
  - ◆ **Timing attack: no extra sensitivity due to protection in synchronous circuits (at least for many classical codes)**
  
- **Concerns**
  - ◆ **Error assumptions – choice of code, and consequences on implementation costs**
  - ◆ **Impact of the algorithm (cipher): are prediction rules available?**
  - ◆ **Power attacks: effect of the added checking bits ?**
  - ◆ **Electromagnetic attacks: derived from the power consumption**

# Choosing the proper EDC

Courtesy P. Maistri

Cipher	$\oplus$	$\wedge, \vee$	$+, -$	$\times$	Sbox	Rot	Sh	Perm	$\times \bmod G(x)$
Camellia	✓	✓			✓	✓		✓	
DES	✓				✓			✓	
IDEA	✓		✓	✓				✓	
MARS	✓		✓	✓	✓	✓		✓	
RC5	✓		✓			✓		✓	
RC6	✓		✓	✓		✓		✓	
Rijndael	✓				✓			✓	✓
Serpent	✓				✓	✓	✓		
Twofish	✓		✓		✓	✓		✓	✓

Cipher	Suggested code	Cipher	Suggested code
AES	Parity, per byte	RC5	Parity or residue
DES	Parity	RC6	Residue
IDEA	Residue, but expensive	Serpent	Parity, per byte
MARS	Residue, but expensive	Twofish	Parity, per byte



# Concurrent error detection schemes – AES (1)

---

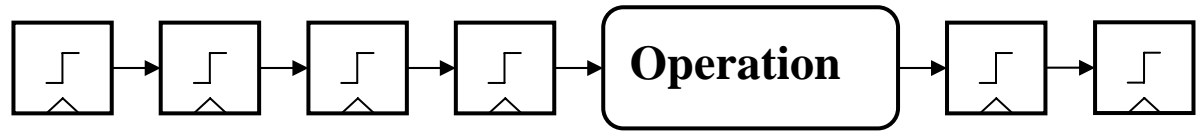
- **Based on spatial redundancy – e.g. circuit duplication**
  
- **Based on information redundancy:**
  - ◆ **Parity Code**
    - **Single parity bit associated to each byte of the state, computed in parallel to encryption (Bertoni et al., TC 2003)**
      - » Predicting the code through the non-linear layer is expensive
    - **Single parity bit associated to the whole state (Karri et al., 2003)**
      - » Cheaper, but also less robust
  - ◆ **Non-linear cubic codes (Karpovsky et al., DSN 2004)**
    - **Very robust, but really expensive...**

# Concurrent error detection schemes – AES (2)

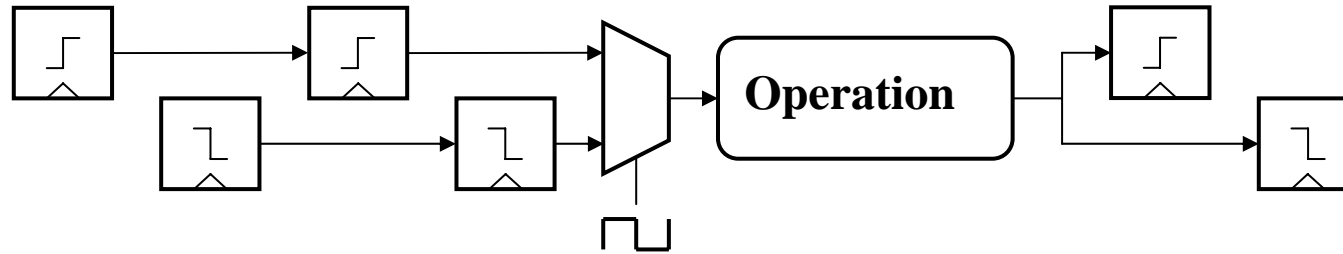
---

- **Based on temporal redundancy:**
  - ◆ **Computation of the inverse process (e.g., decryption) with additional (possibly existing) hardware (Karri et al., 2001)**
  - ◆ **Computation of the inverse process with the same hardware, for involution ciphers only (Joshi et al., CHES 2004)**
  - ◆ **Repetition of the same process, exploiting a (possibly existing) pipeline (Wu and Karri, DFT 2001)**
    - **Pipeline cannot be filled with consecutive blocks, since they have to be encrypted sequentially for security reasons ...**
    - **... then use the empty stages of the pipeline to reissue the last computation and check if the results are different**
  
- **DDR: other approach to temporal redundancy...**

# Double-Data-Rate computation



4 clock cycles to compute *Operation* for all input bytes



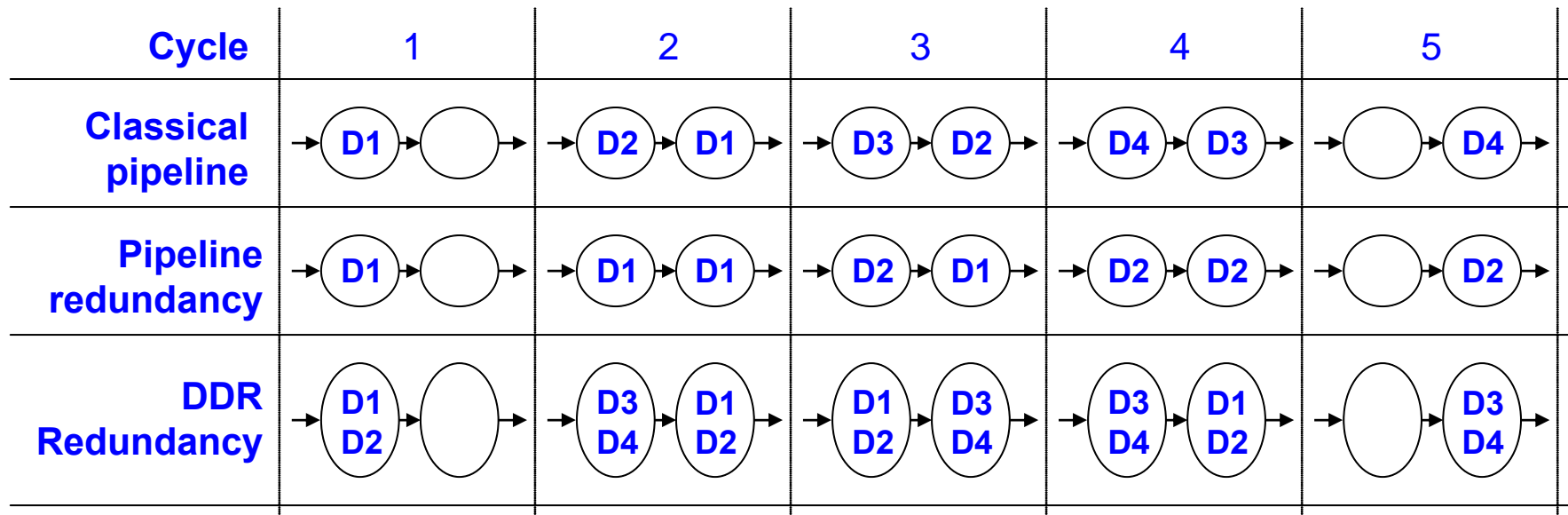
2 clock cycles to compute *Operation* for all input bytes

- ✓ Twice the throughput at the same frequency
- ✓ Small area overhead for DDR logic
- ✓ Increased parallelism
- ✗ More complex routing, thus lower max frequency
- ✗ Error detection requires additional overhead
- ✗ Design may require synchronization "bubbles"

+ temporal separation of duplicated computations

# DDR vs. classical pipeline redundancy

Case of "long" perturbations:



# Protections beyond DDR: controller

---

**Reduce the number of possible targets**

**Simplify the controller removing redundant registers, which store signals that could be instead computed on the fly**

**Protect sensitive targets**

**Specific registers are duplicated to ensure correct behavior (e.g., counters, state registers)**

**Validate state encoding**

**If any FSM falls into an unused state encoding, computation stops and they both return to the reset state**

**Verify state transitions**

**If an FSM performs an erroneous transition (e.g., Idle > Output), the error signal is raised and the machines go back to their reset state**



# Injection campaign

---

## □ Location

- ◆ Linear multiplication layer
- ◆ Barrel shifter (row rotation)
- ◆ Non-linear substitution layer: inner and outer stage
- ◆ Control unit
- ◆ AES is highly regular: only one target element for each data path location

## □ Timing

- ◆ Every computation clock cycle
- ◆ Input or output phase were not considered
- ◆ **From 1 up to several clock cycles (twice the length of a round)**

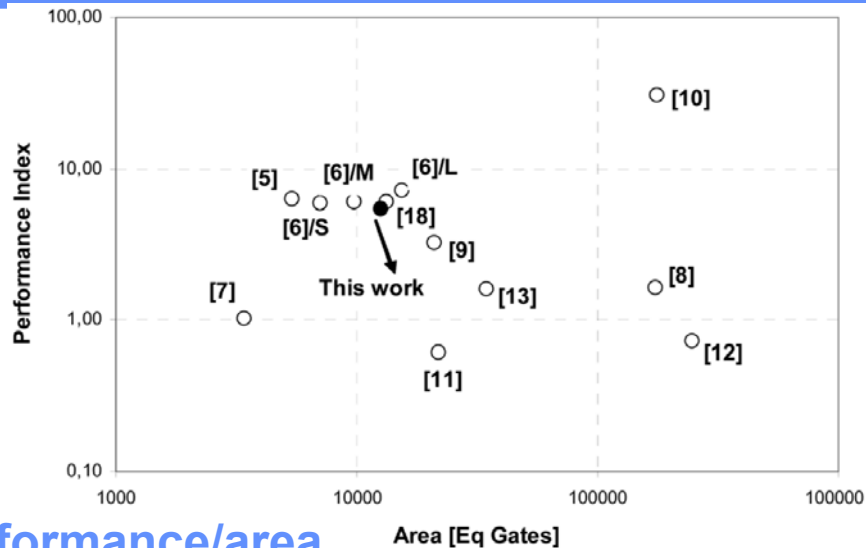
## □ Value

- ◆ According to the laser-injected fault model, the error value is not controllable
- ◆ **Exhaustive search of all error values is carried on for each targeted area (e.g., all byte values for a byte target)**

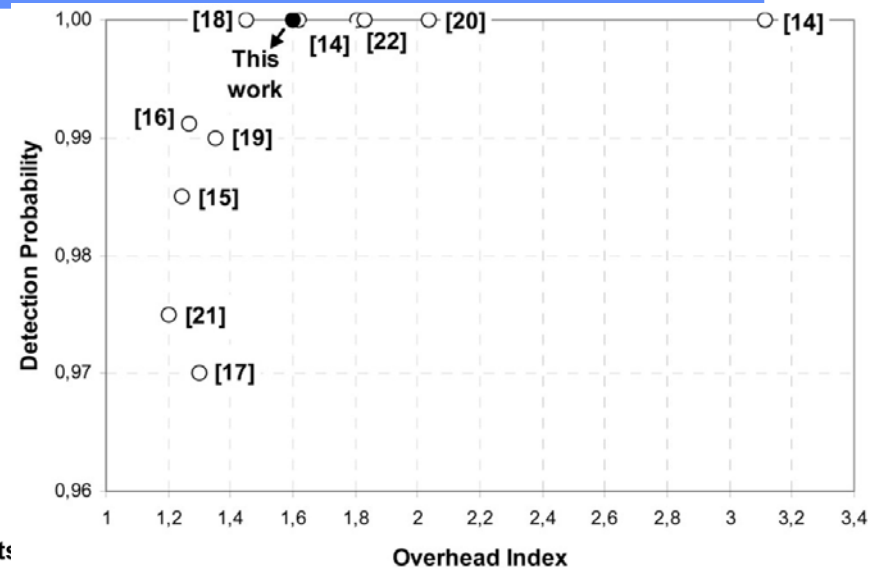




# AES DDR – Results [TC'08]

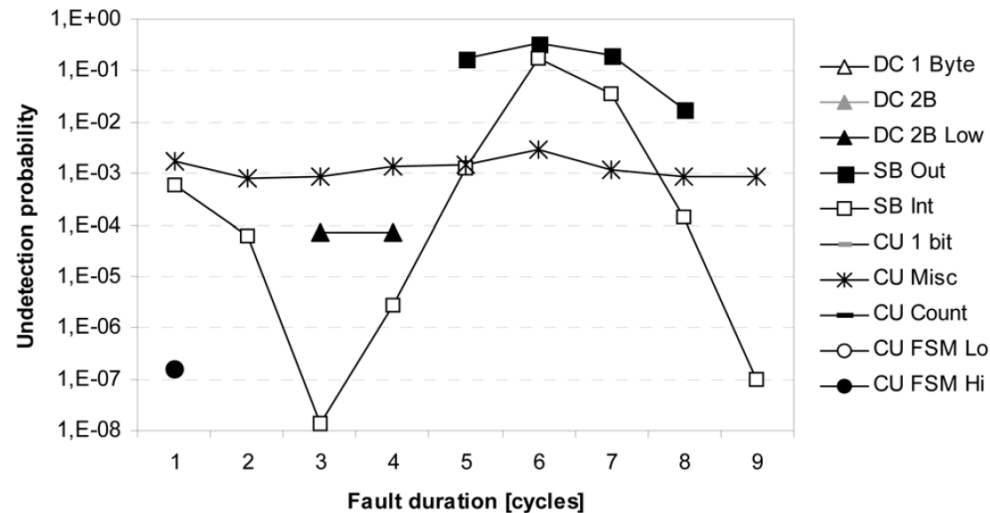


Performance/area characteristics



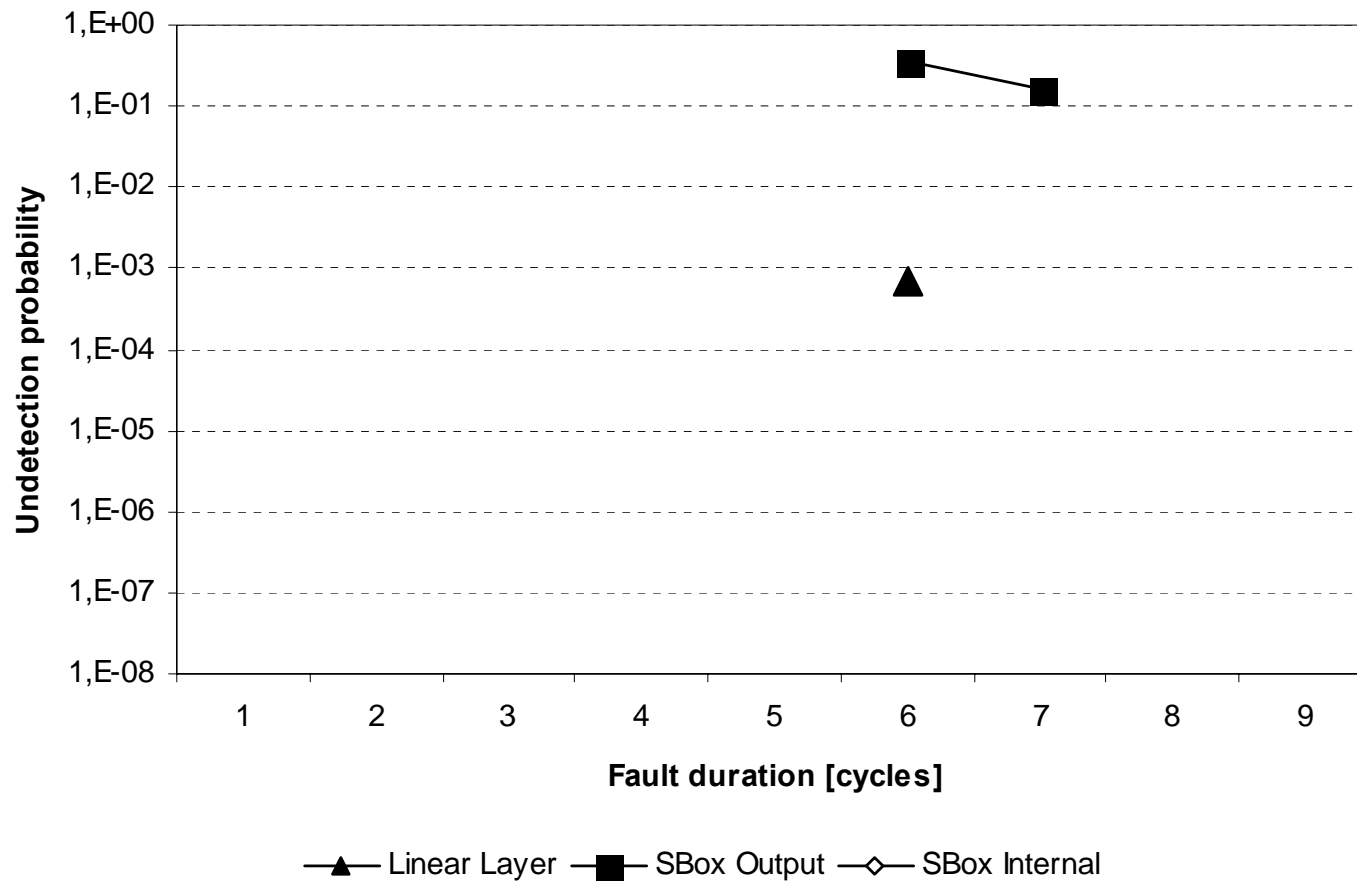
Detection vs. overheads

Undetected faults vs. duration

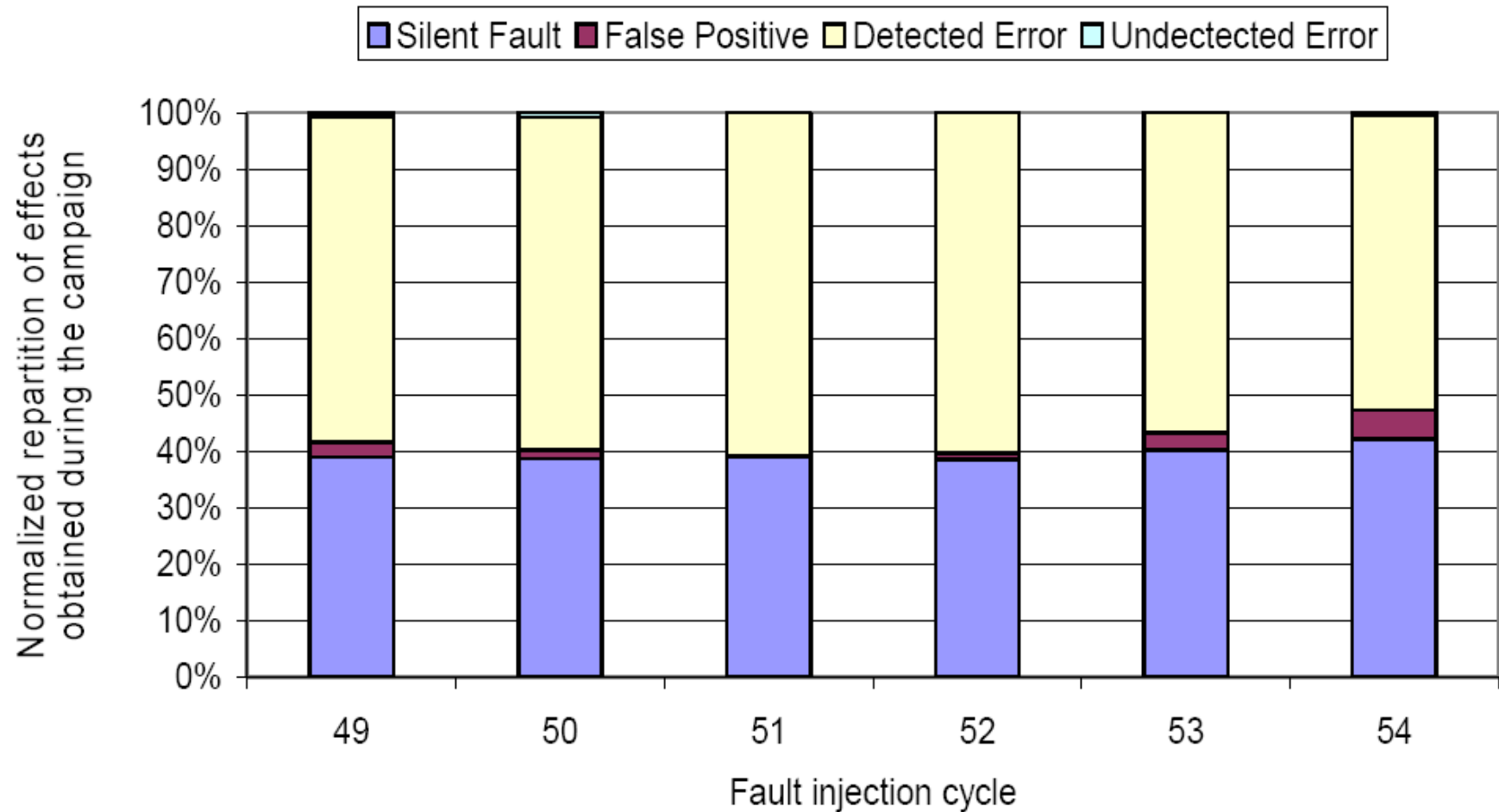


# An improved counter-measure

- Taking into account remanent (or permanent) faults  
=> resource reallocation between normal and verification rounds



# Laser attacks on last DDR protected version



# False positives - Warning !!!

---

- **Today, in many cases, availability is not addressed in security policies**
  - ◆ **Memory burning after attack**
  - ◆ **Other "destructions" of sensitive data making the system impossible to use**
  
- **Potential consequences**
  - ◆ **Embedded systems: OK if the user is the attacker (e.g., corruption of his own set-top-box)**

**BUT**

- ◆ **Deny-of-service attacks**
- ◆ **Terrorism ...**
- ◆ **Degraded services**
- ◆ **...**



# Overview of efficiency vs. overheads [P. Maistri, IOLTS 2011]

Type	Description	Permanent Faults		Multiple transient	Area overhead	Throughput reduction
		Single	Multiple			
Hardware	Duplication	100%	☺	☺	> 100 %	-
	Partial red	☺	☺	☹	25 – 35 %	-
Information	Byte/Word Codes	100 %	> 99 %	50 – 99 %	20 – 30 %	3 %
	Block Codes	98 %	☹	☹	18 – 24 %	~ 0 %
	Nonlinear code	1 – 2 <sup>-56</sup>	1 – 2 <sup>-56</sup>	☺	77 %	13 %
Time	Process repetition	0 %	0 %	☺	~ 0 %	50 %
	Involution ciphers	100 %	☺	☺	~ 0 %	50 %
	SLICED	100 %	☺	☺	~ 0 %	64 %
	Pipeline Red	0 %	☹	☺	50 %	18 %
	DDR Red	0 %	☹	☺	36 %	15 – 55 %
	DDR++	> 90 %	☺	☺	~ 36 %	~ 15 – 55 %
Mixed	Inverse Operation	100 %	☺	☺	19 – 38 %	23 – 61 %
	Inv Pipeline	100 %	☺	☺	-21 – 24 %	~ 25 %



---

# **Influence of design style**

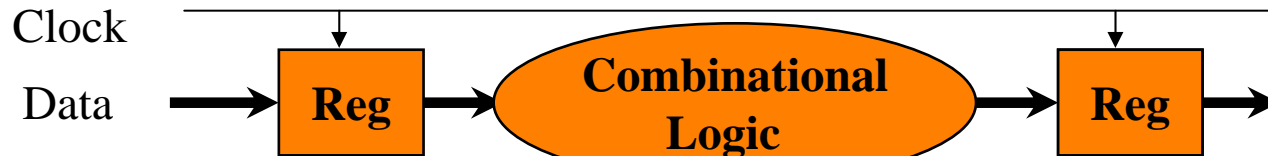
## **Example:**

### **synchronous vs. asynchronous logic**

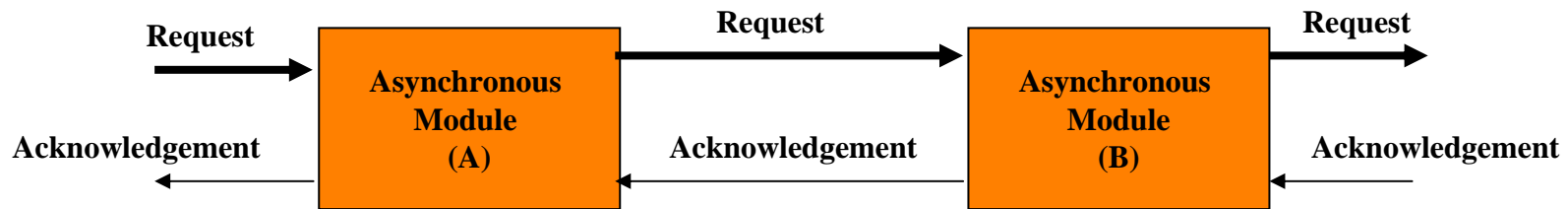


# Asynchronous Logic: Quasi Delay Insensitive circuits

---



**Basic structure of a synchronous circuit**

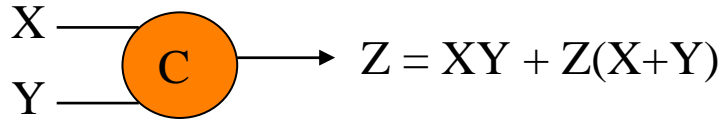


**Handshake-based communication between modules (QDI)**

**Intrinsically more robust to perturbations (delay faults)  
and SCA (better current profile)**

# QDI memory cell: the Muller gate (C element)

---



**Truth table:**

X	Y	Z
0	0	0
0	1	$Z^{-1}$
1	0	$Z^{-1}$
1	1	1

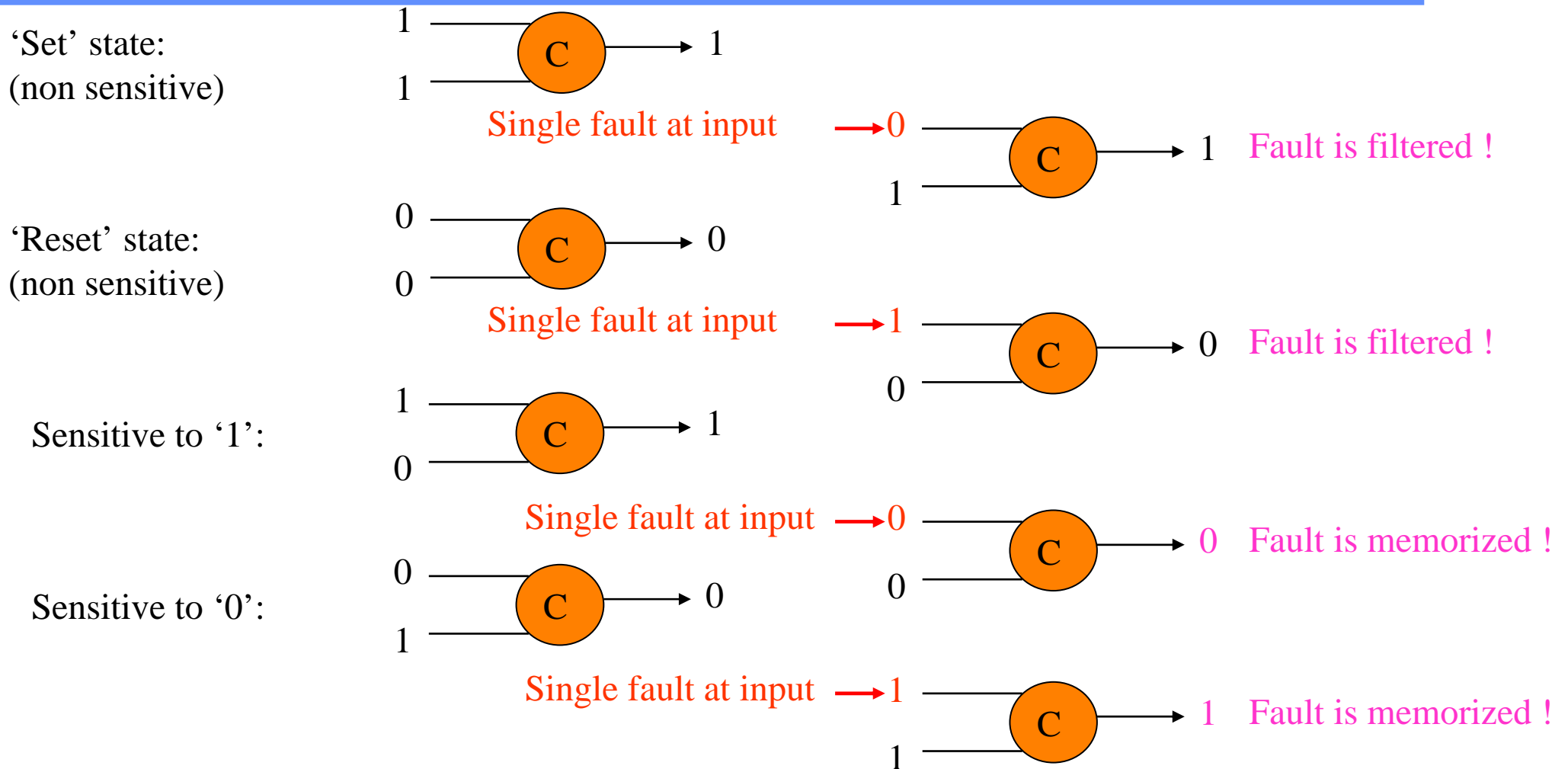
This gate is used to implement both the data paths and the protocol specification.

**QDI circuits sensitivity criterion =>**

**Ability of Muller gates to memorize/filter a transient fault**



# QDI circuits: specific sensitivity criterion



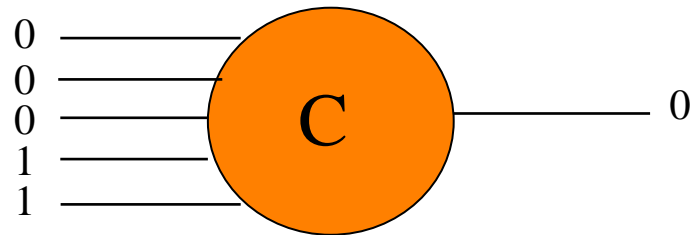
Y. Monnet, M. Renaudin, R. Leveugle, "Asynchronous circuits sensitivity to fault injection",  
 10th IEEE International On-Line Testing symposium, Funchal, Madeira, Portugal, July 12-14, 2004, pp. 121-126



# Generalization of the sensitivity criterion

---

N-input Muller gate: sensitivity to  $[1..M]$  erroneous inputs



A “3-sensitive to 0” Muller gate

# QDI circuits: specific protections

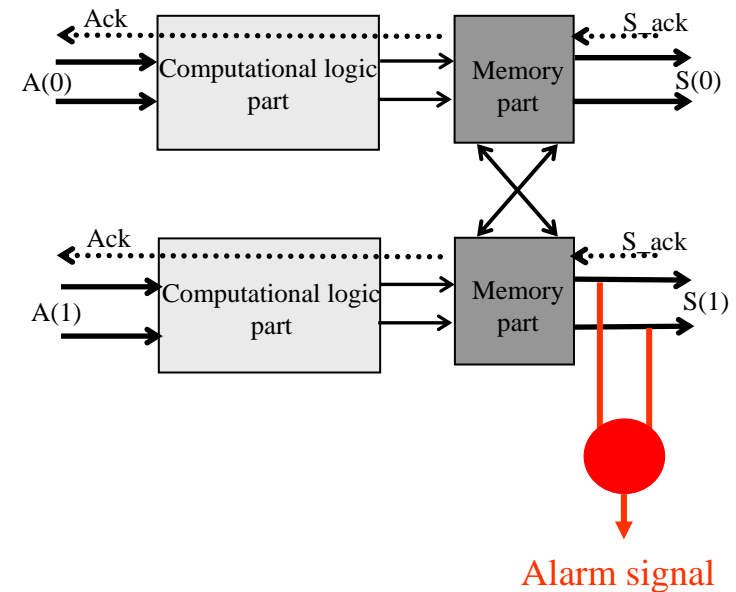
## □ Objective :

- ◆ Protecting logic blocs against transient faults
- ◆ Detecting wrong code generation

## □ Rail synchronization technique

- ◆ Improves the tolerance against transient faults
- ◆ Improves the wrong code detection when combined with alarm cells

## □ Alarms cells



Y. Monnet, M. Renaudin, R. Leveugle, N. Feyt, P. Moitrel, F. M'Buwa Nzengué, "Practical evaluation of fault countermeasures on an asynchronous DES cryptoprocessor", 12th IEEE International On-Line Testing symposium, Como, Italy, July 10-12, 2006, pp. 125-130

# Conclusions

---

- ❑ **Take care of hardware/software implementation characteristics when security is a constraint ... or your system will be at risk !**
- ❑ **Don't rely on too simple assumptions (e.g. single bit flips)**
- ❑ **Finding clever global solutions is a hot topic for research**

