



Esisar
VALENCE



Fault Injection Attacks

David Hély
Amir-Pasha Mirbaha
Ihab Alshaer



November - December 2021

Summary

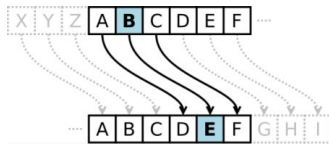
- Definition
- Methods
- Some examples
- Some Countermeasures

Recall

Cryptography: Study and practice of secret message writing methods

Purpose: To make the messages unreadable to anyone who is not the intended recipient

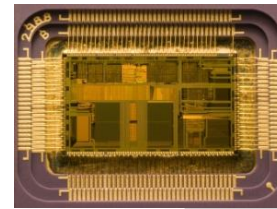
3 historical periods:



Handwork



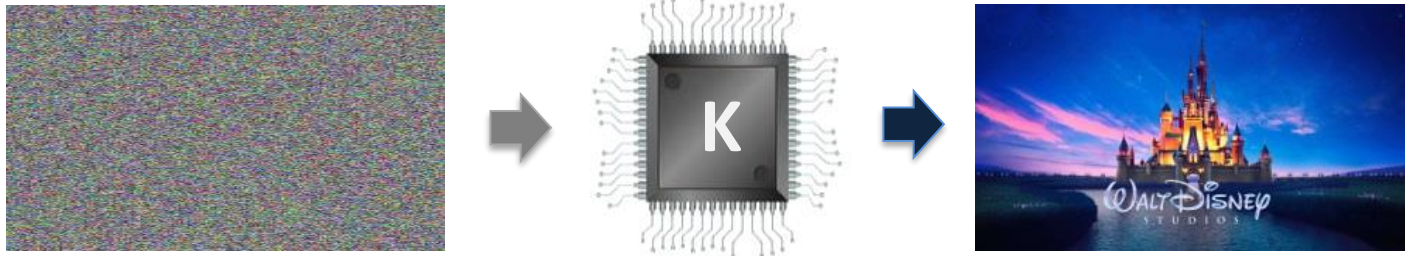
Electromechanic



Electronic

Recall

- Case of a cryptographic module
 - Video decoder
 - Input: encrypted video
 - Output: decrypted video
 - Secret: cryptographic key



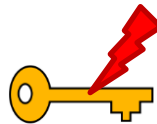
$$CIPHER = f(PLAINTEXT, K) \text{ and } PLAINTEXT = f^{-1}(CIPHER, K)$$

Kerckhoffs' principle

Public algorithm, security is based on the confidentiality of secrecy.

Secret : 

How to find the secret?

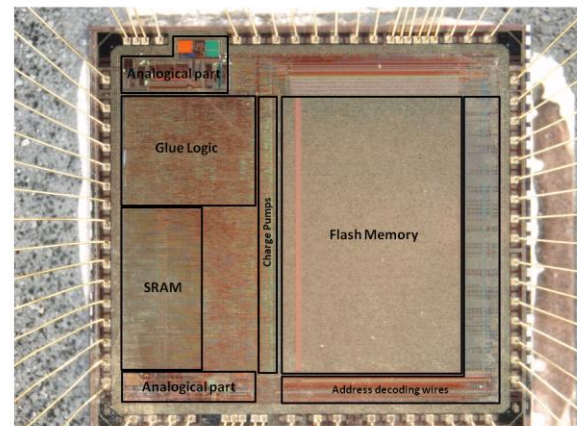
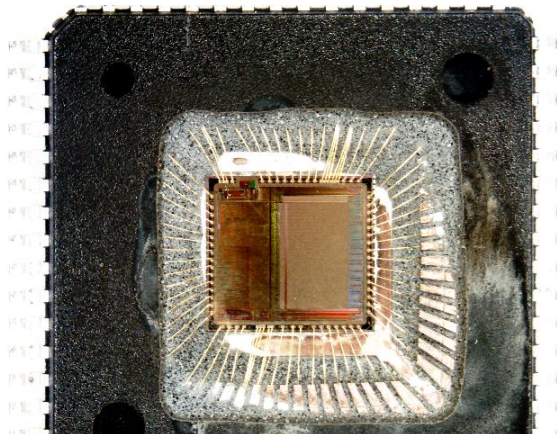


How to attack the key?

Searching vulnerabilities

Vulnerabilities of cryptographic circuits

- Cryptanalytic attacks:
 - Exhaustive search and mathematical attacks
- Hardware attacks:
 - Side-Channel Analysis: Observation of physical parameters
 - Fault Injection: Circuit disturbance or modification of the key
 - Invasive attacks: Opening and modifying the component



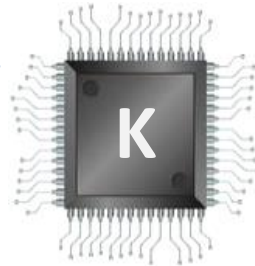
Side-Channel Analysis

- Side-Channel Analysis: Observation of physical parameters

Power Consumption Analysis

Calculation Time Analysis

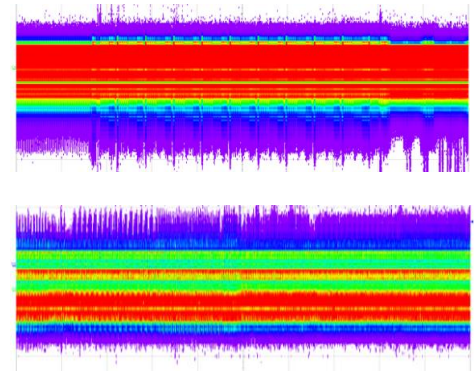
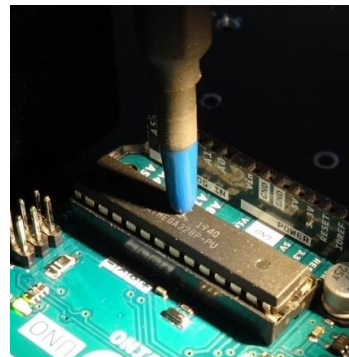
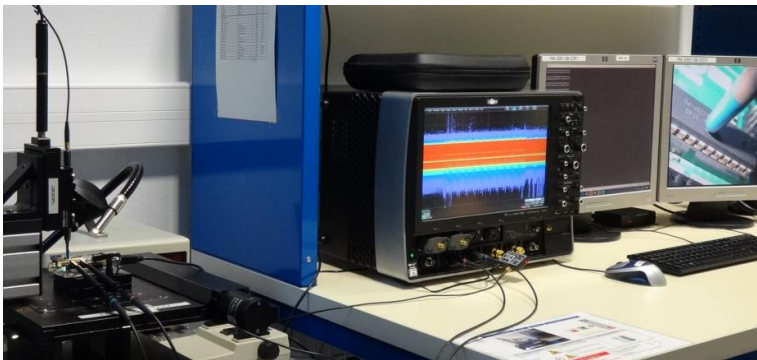
(Heat Dissipation)



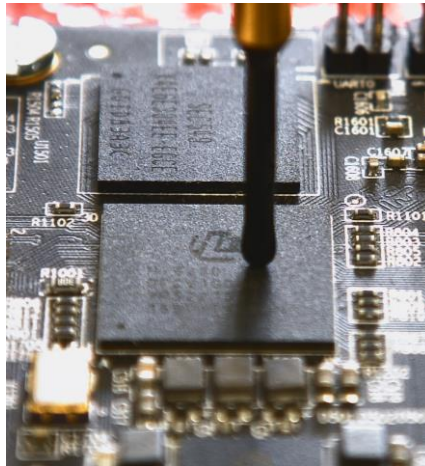
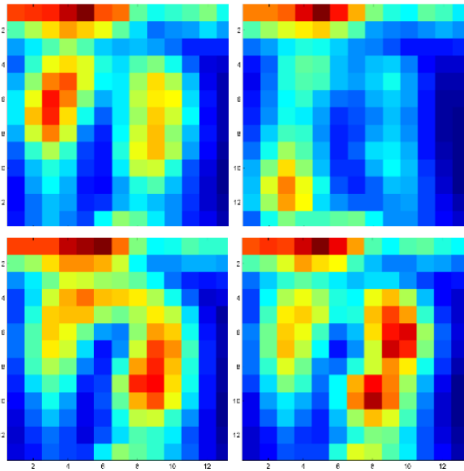
Electromagnetic Field Analysis

...

(Photon Emission)

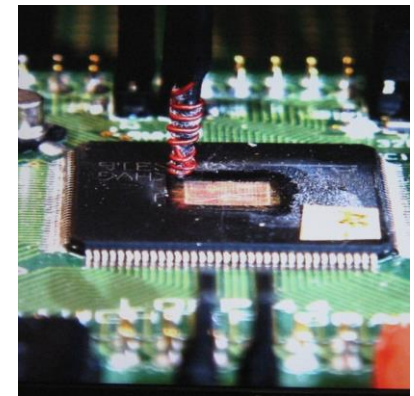
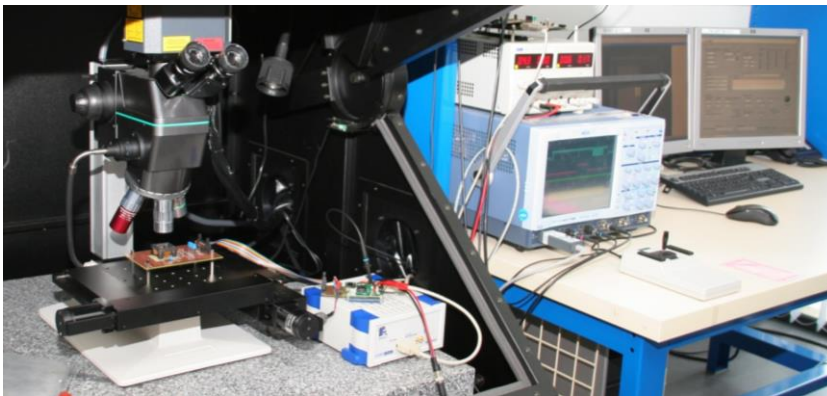
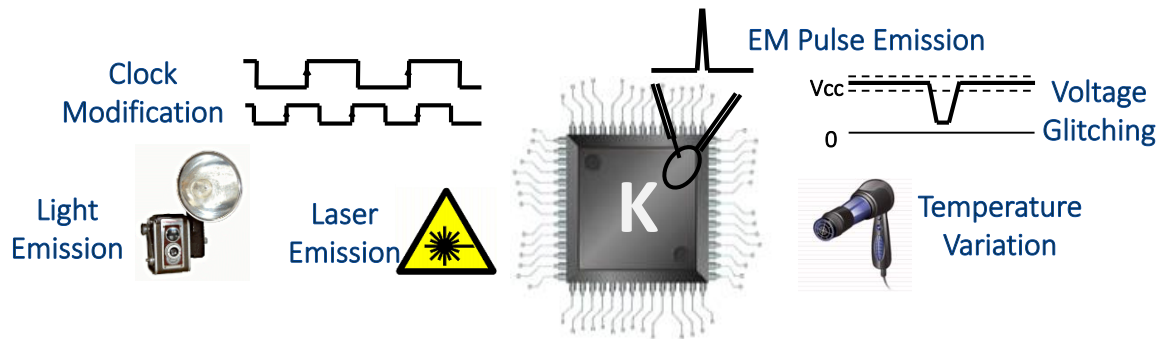


Side-Channel Analysis

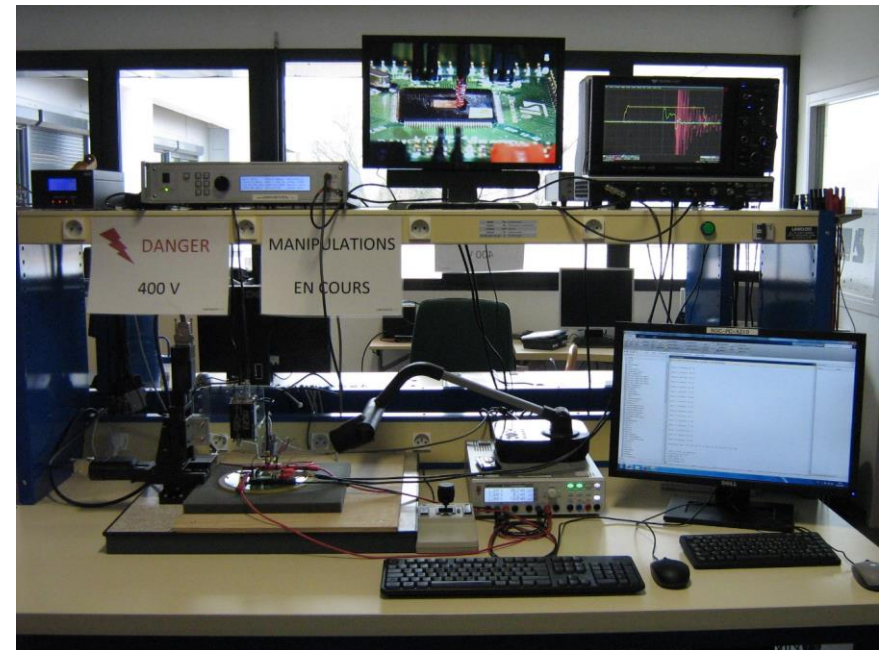
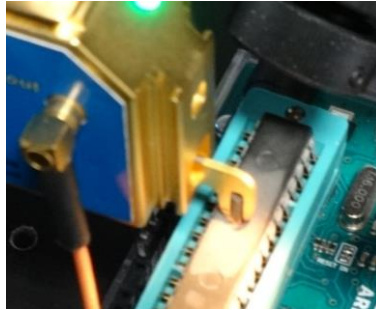
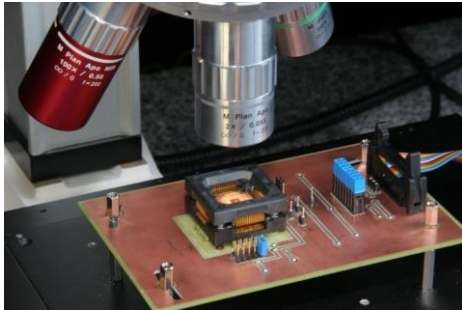


Fault Injection

- Fault Injection: Circuit disturbance or modification of the key



Fault Injection



Fault Injection

- Definition:
 - The so-called fault injection, whatever it works with (rays, glitch, temperature) aims at disturbing the normal flow of execution of the running program, or at modifying the content of memories or registers. One generally distinguishes transient faults from permanent faults.

Fault Injection

- Induce errors in the device
- Perform encryption with and without presence of fault
- Recover sensitive information
- Large number of devices in the market very vulnerable to these attacks
- Attackers are getting more and more sophisticated
- Attack Equipment relatively cheap

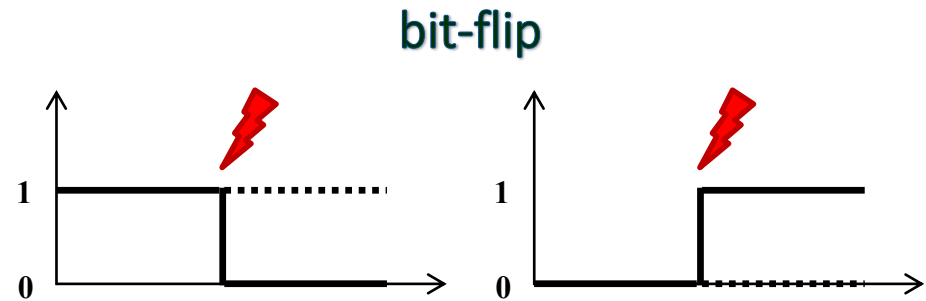
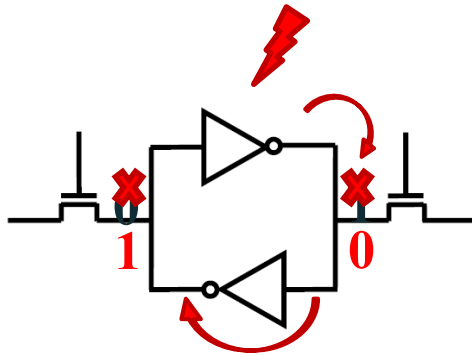
Methods of Fault Injection

- Variation in Supply
 - May cause a processor to misinterpret or skip instructions
- Variations in the External Clock
 - May cause data misread
- Temperature
 - Manufacturers define upper and lower thresholds within which their circuit will function correctly. Temperature variation outside the bounds can modify RAM values or alter the read and write properties of NVM.

Methods of Fault Injection


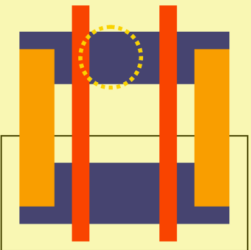

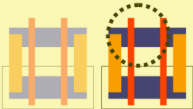



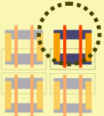
- White Light
 - The current induced by photons can be used to induce faults
- Laser
 - The effect is similar to white light, the advantage is directionality that allows to precisely target a small circuit area
- Electromagnetic Impulsion
 - May cause a processor to misinterpret or skip instructions

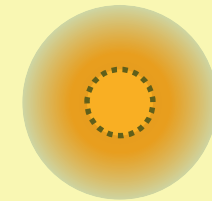
Fault Injection



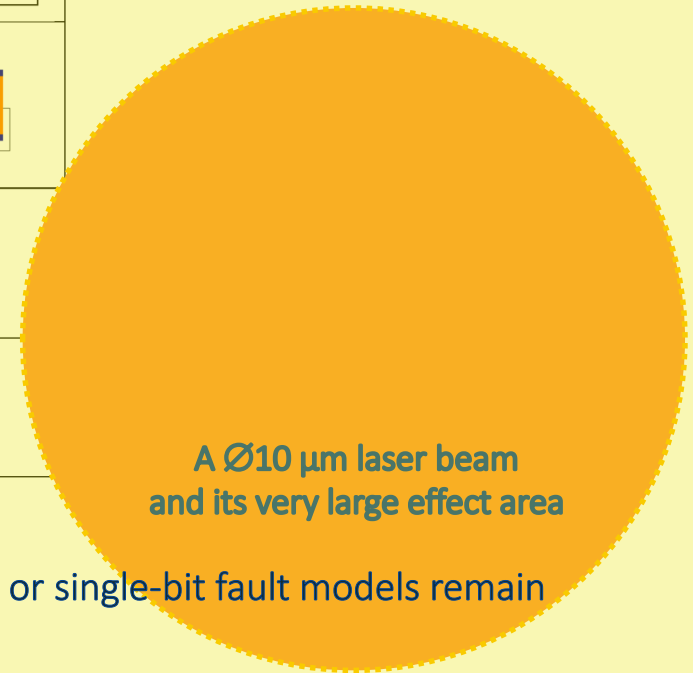
effet du transitoire \Rightarrow bit-flip \Rightarrow injection de faute

Laser Fault Injection

	Transistor	SRAM Cell
0,35 μm		
0,13 μm		
90 nm		
65 nm		



A $\varnothing 1\mu\text{m}$ laser beam
and its effect area



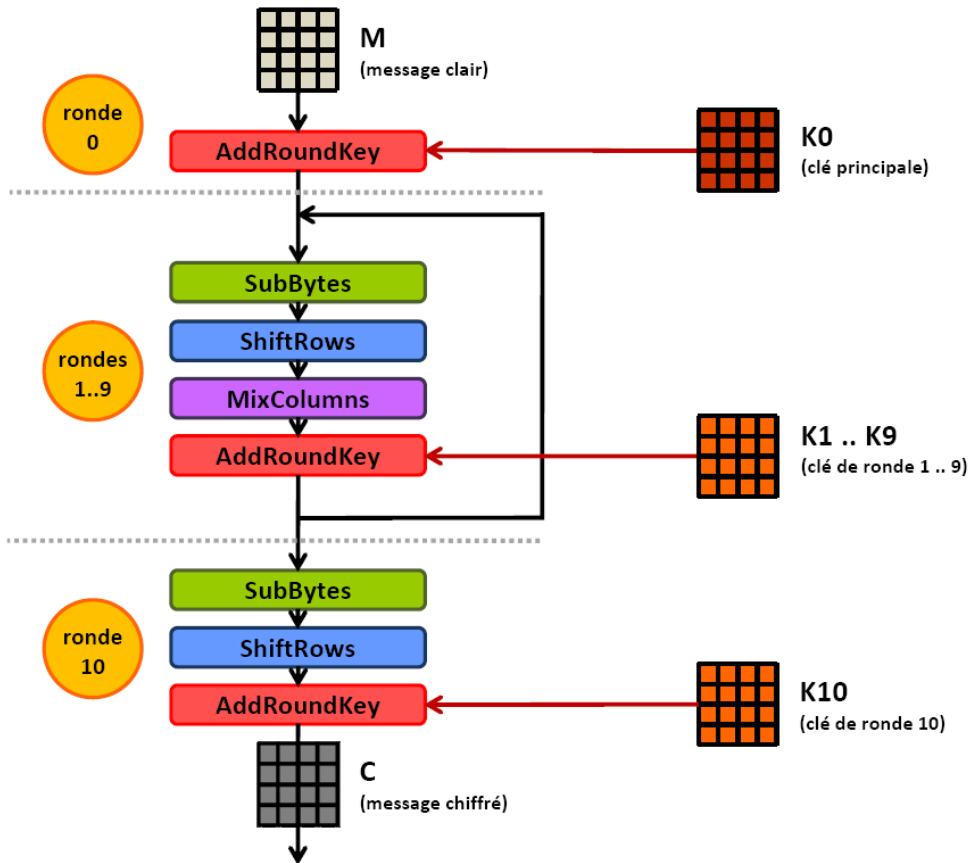
A $\varnothing 10\mu\text{m}$ laser beam
and its very large effect area

With technological developments, will single-byte or single-bit fault models remain realistic?

Fault Injection Parameters

- Control on the fault location
 - Which part of the system?
- Control on the timing
 - Which trigger?
- Number of affected bits
 - Single-bit, single-byte, random
- Duration
 - Transient, permanent, destructive
- Probability
 - Success probability
- Repeatability
 - Is the same fault reproducible?

Advanced Encryption Standard (AES)

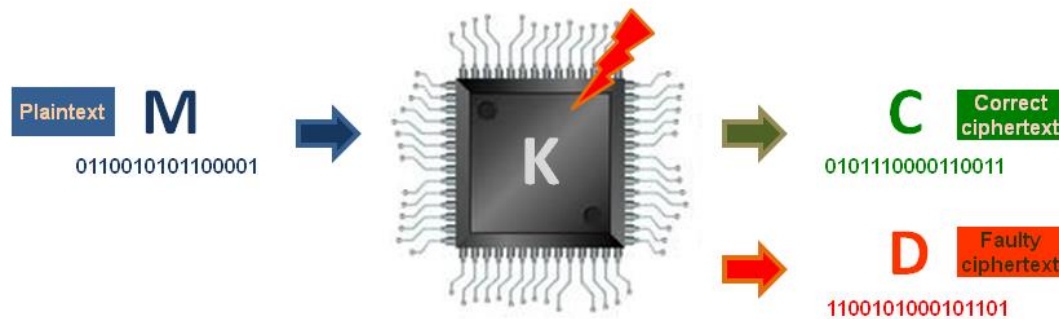


- A symmetric cryptography algorithm
- 16-byte blocks for input and output
- AES-128 bits: 10 rounds (after an initial round) with a 128-bit secret key

Differential Fault Analysis (DFA)

Principles of a DFA:

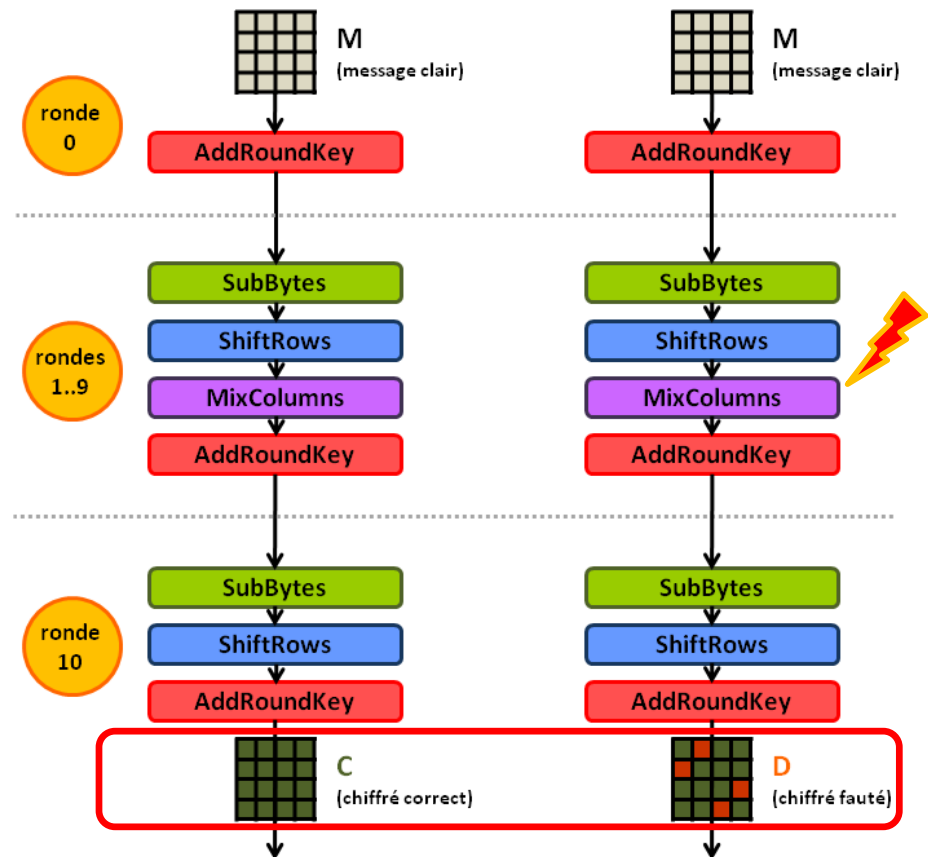
- Injection of faults during cryptographic calculations



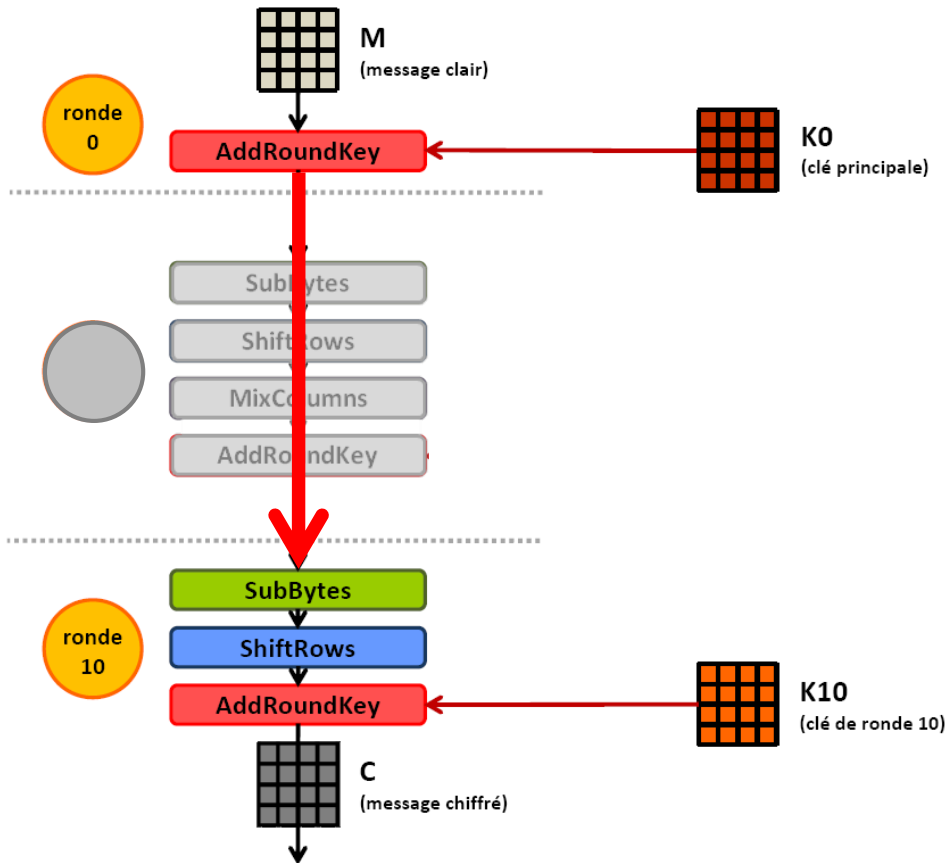
- Extraction of information on the key by comparison of faulty and correct encryptions
- Restricted fault model: single-byte or single-bit fault injection

Single-Byte DFA: Piret & Quisquater

- Classic attack based on the single-byte fault model
- Comparison of the corresponding incorrect and correct ciphers to deduce part of the key



Some Fault Injection Scenarios on the AES



START

$C \leftarrow M$

$C \leftarrow C \oplus K$

FOR ($RC \leftarrow 1$; $RC < R_{\max}$; $RC++$)

{

$C \leftarrow \text{SBOX}(C)$

$C \leftarrow \text{SR}(C)$

$C \leftarrow \text{MC}(C)$

$C \leftarrow C \oplus K_{RC}$

}

$C \leftarrow \text{SBOX}(C)$

$C \leftarrow \text{SR}(C)$

$C \leftarrow C \oplus K_{R_{\max}}$

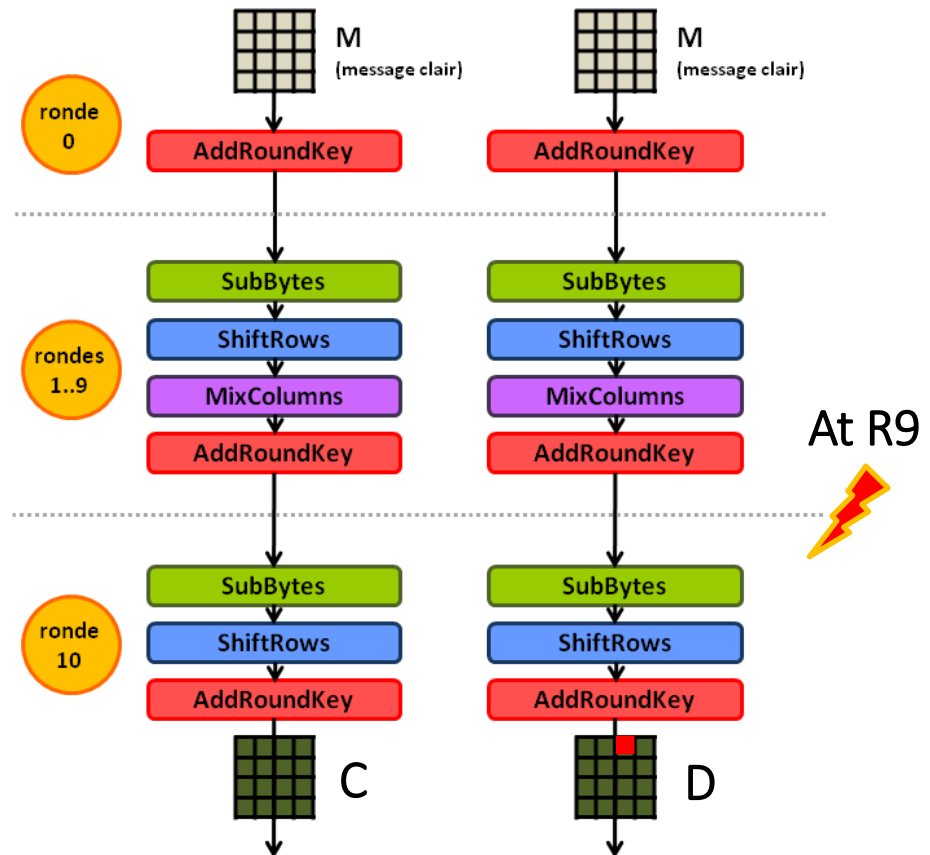
END

Round Reduction by H. Choukri and M. Tunstall, 2005

Some Fault Injection Scenarios on the AES

Single-Bit and Single-Byte Attacks by Ch. Giraud

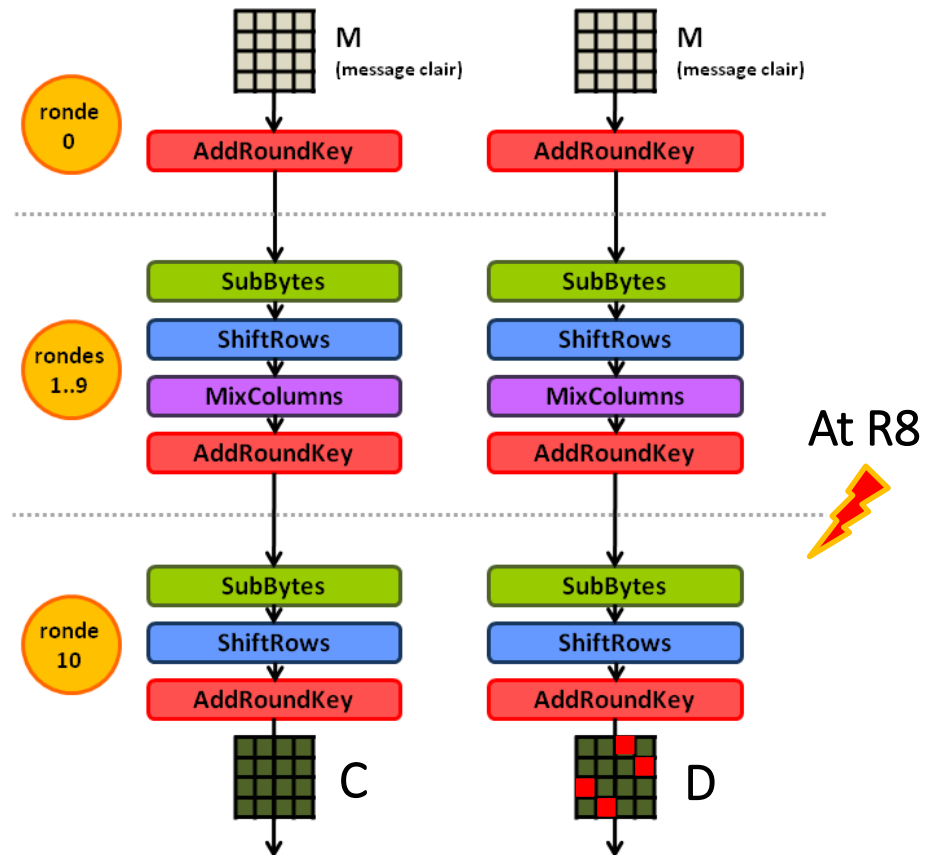
- Two classical attacks, based on the single-bit and single-byte fault model
- Comparison of the corresponding faulty and correct ciphers to deduce part of the key



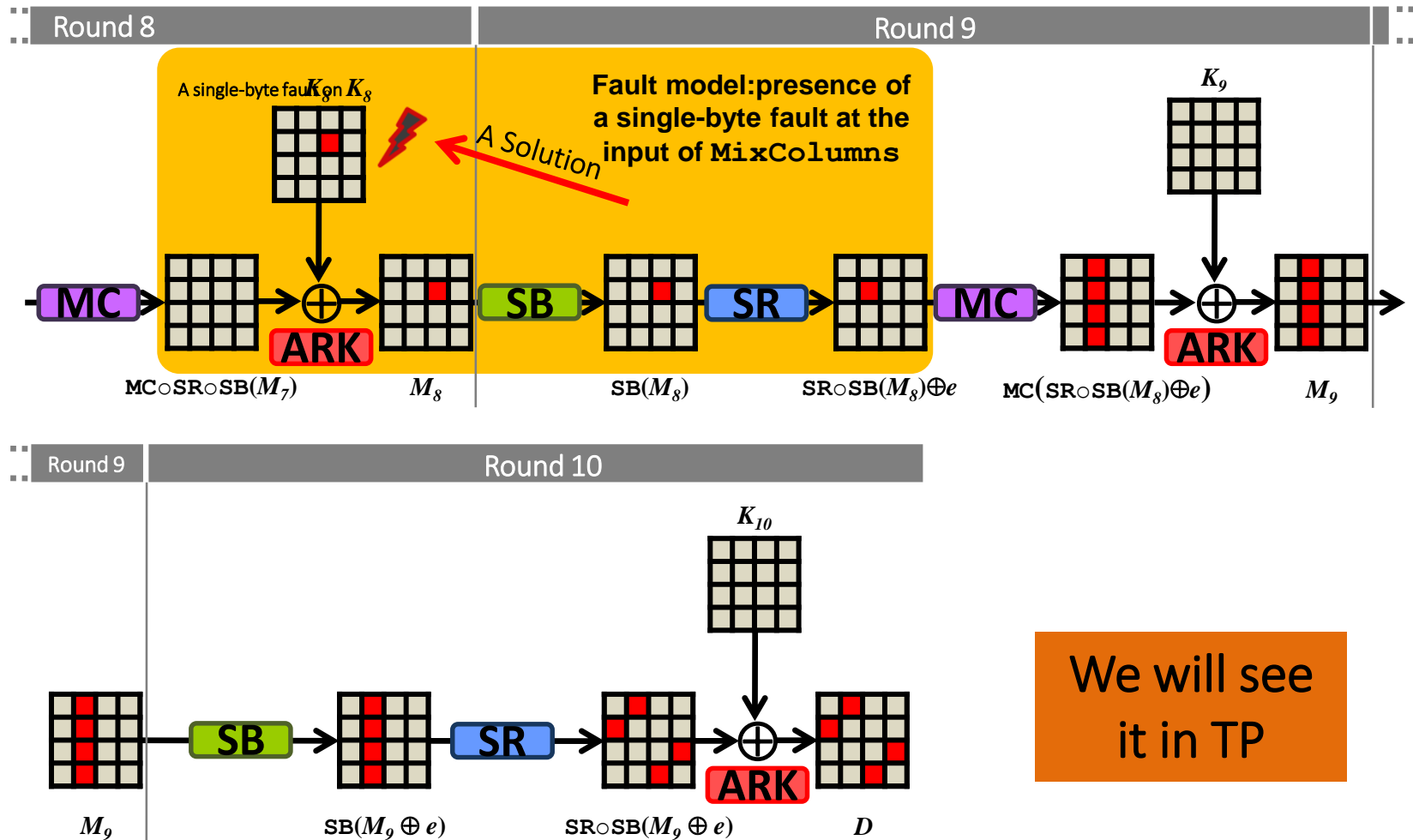
Some Fault Injection Scenarios on the AES

Single-Byte Attack by Piret and Quisquater

- Classical attack based on the single-byte fault model
- Comparison of the corresponding faulty and correct ciphers to deduce part of the key



Single-Byte DFA: Piret & Quisquater



Fault Injection by Clock Glitching

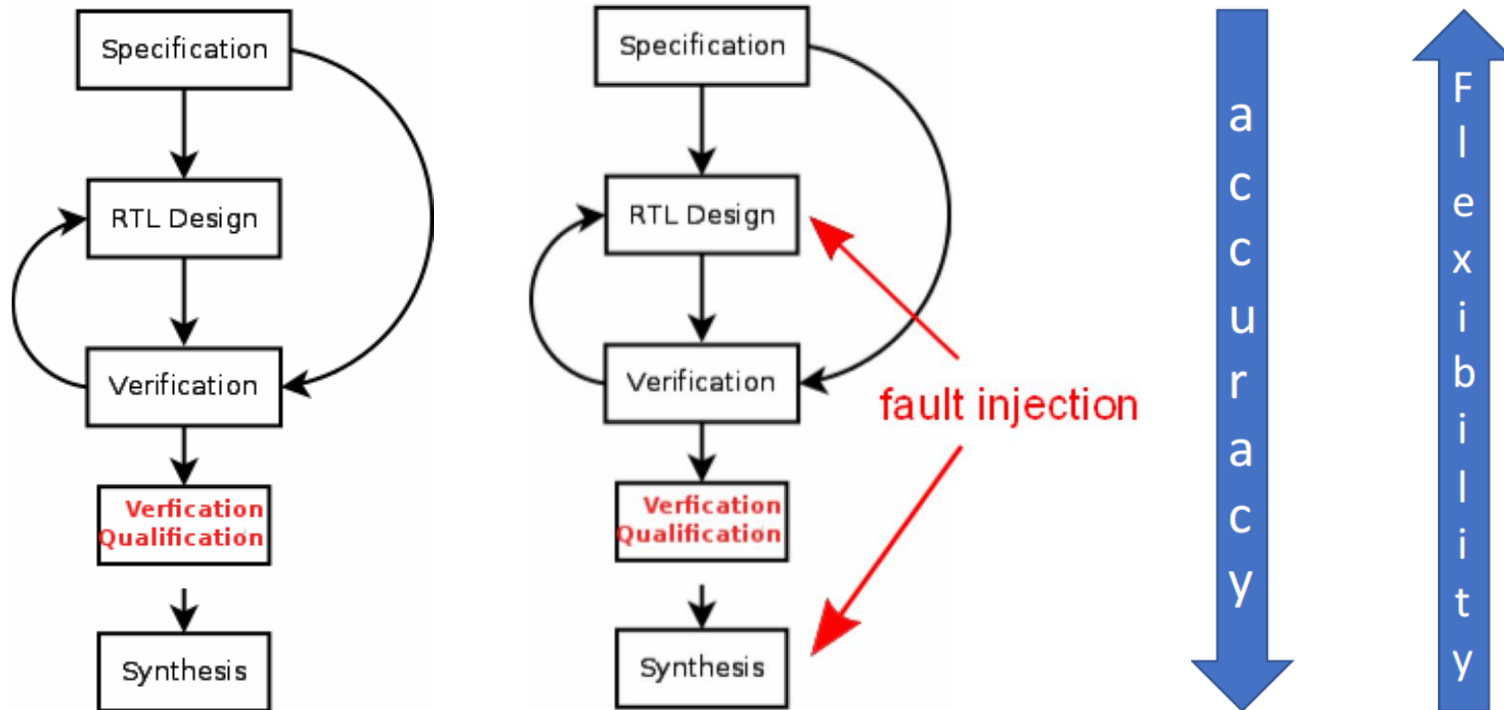
- Presentation by Ihab ALSHAER

We will see
it in TP

Fault Injection Evaluation?

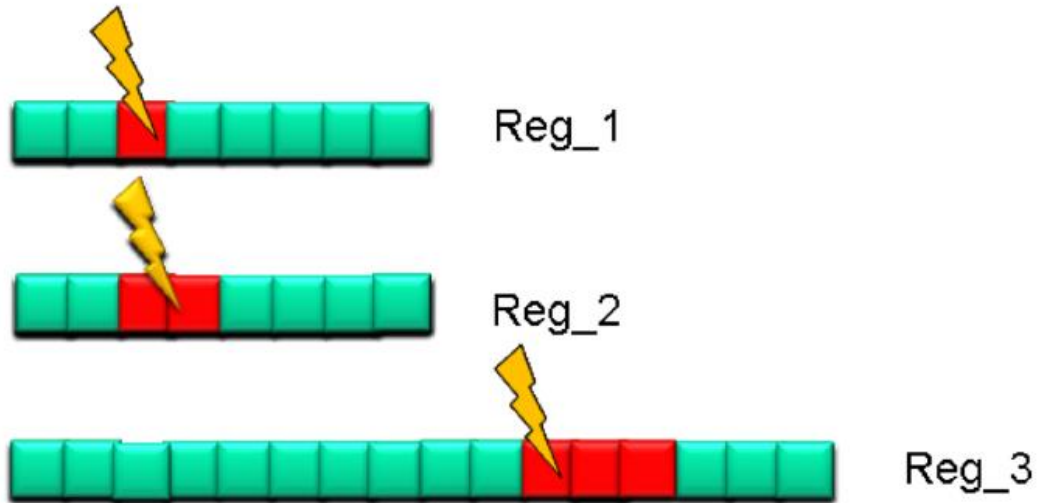
- What do we want to evaluate:
 - A circuit?
 - i.e. a digital circuit designer wants to evaluate the vulnerabilities of the circuit against FI.
 - A program?
 - A software developer wants to evaluate how the program will behave in case of FI.
- When do we want to evaluate?
 - At design time?
 - How do we model the FI process?
 - On which system description do we perform the evaluation?
 - On the deployed system?
 - Can we emulate FI?
 - Can we use FI equipment?

FI Evaluation of circuit under design

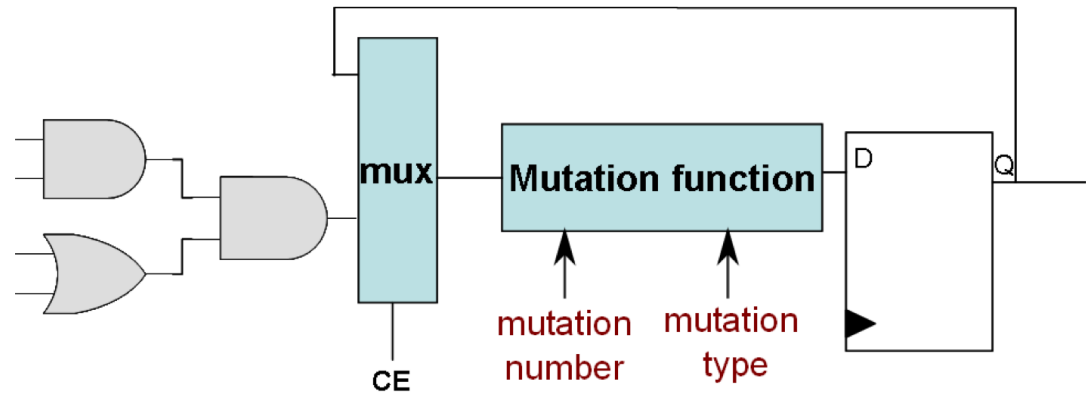
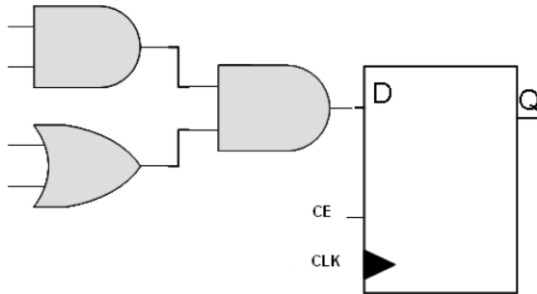


Fault Modeling

- Fault Model:
 - SEU, MEU...
 - Transient...



Fault Modeling Example

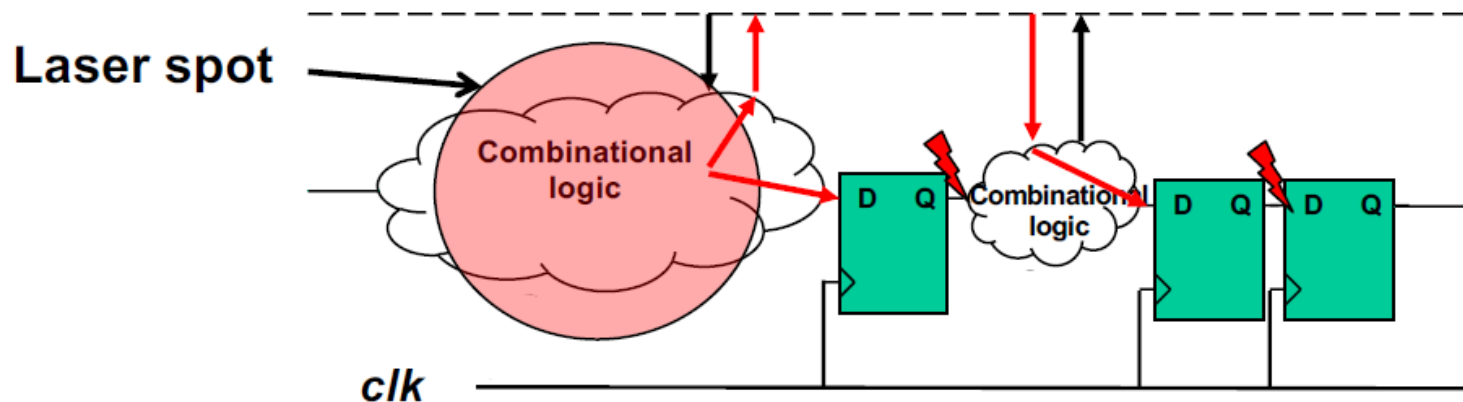


Use Case: Laser Attack

- Semi-invasive attack that uses a light beam to inject faults into semiconductor devices
- Main Characteristics:
 - precise spatial focus
 - accurate synchronization
 - high occurrence probability
- The diameter of the laser spot can be as small as $\varnothing 1\mu\text{m}$
 - a laser spot can hit more than one transistor as the technology evolves

Laser Attacks Effects

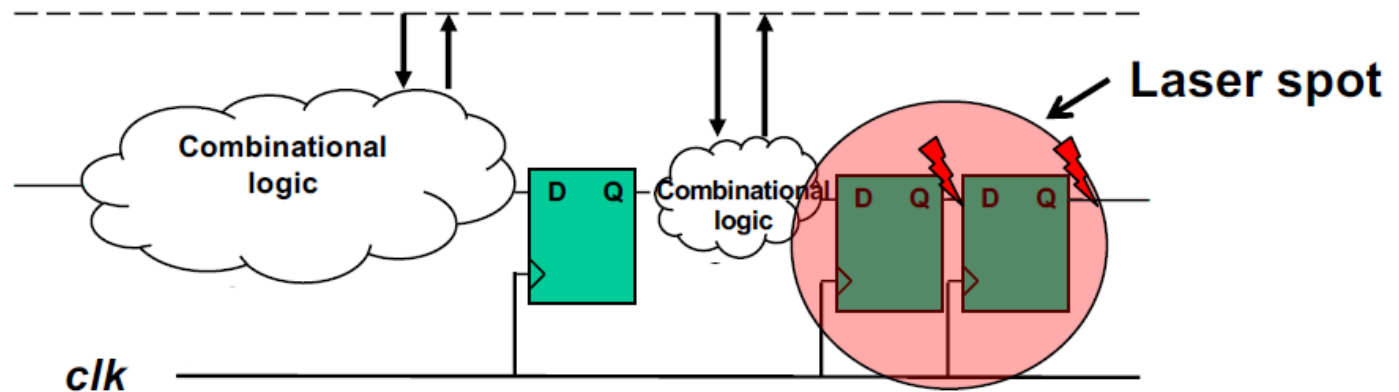
- Laser shot can generate errors in different ways:



Errors are injected into a combinational part and propagated to several FFs

Laser Attacks Effects

- Laser shot can generate errors in different ways:

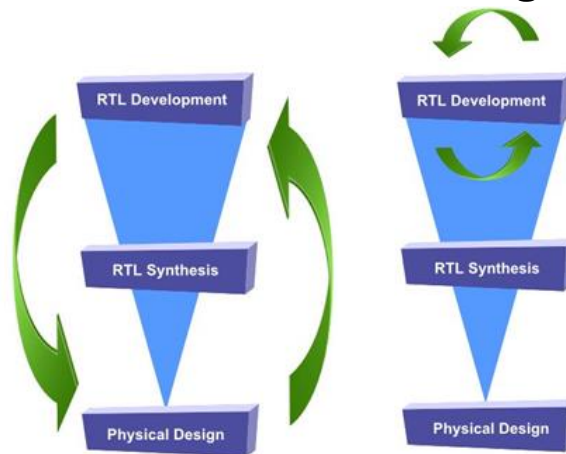


Errors are directly
injected into the
FFs

At RTL level they are modeled by SEU or MEU

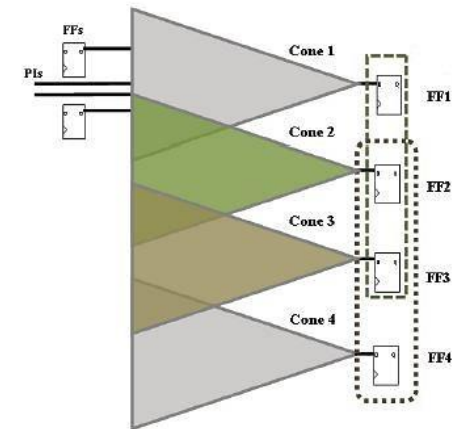
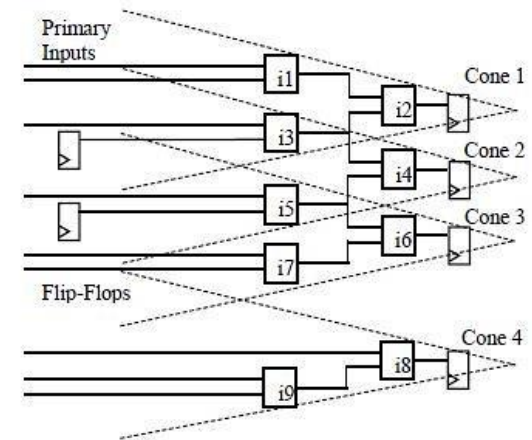
Robustness Evaluation

- Fault injections are performed to validate the detectability of the countermeasures at RTL with:
 - Random fault model
 - Fault samples are generated randomly
 - Laser attack RTL fault model
 - Fault samples are generated from flip-flop sets (locality preserved)
 - Design re-spins are kept at RTL
- Evaluating the efficiency of countermeasures using layout information with:
 - Layout fault model



Laser Attack RTL Fault Model (LAFM)

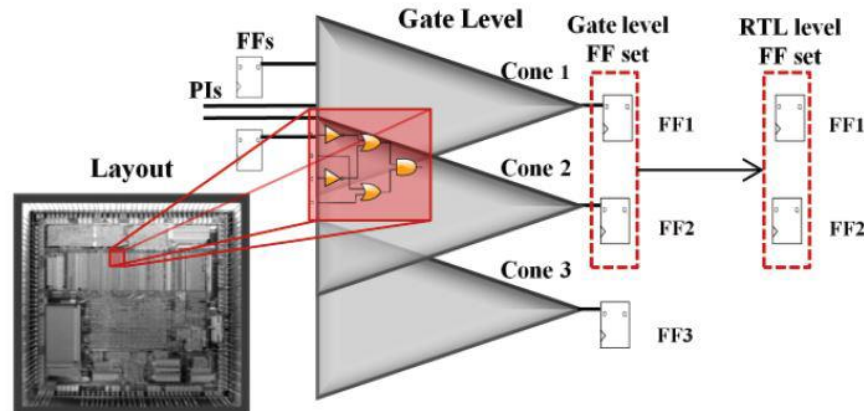
- LAFM is modeling the effects of laser attacks at RTL
 - Cone partitioning of the design
 - Logic cones consists of:
 - Combinational instances
 - Primary inputs
 - Cones are processed by an intersection function
 - Distinguish independent from dependent cones
 - Dependent cones end up in the same set
- The assumption
 - Faults occurring in the instances of a logic cone is possible to propagate to the intersecting cones



During Evaluation MEU are injected only in dependent FF

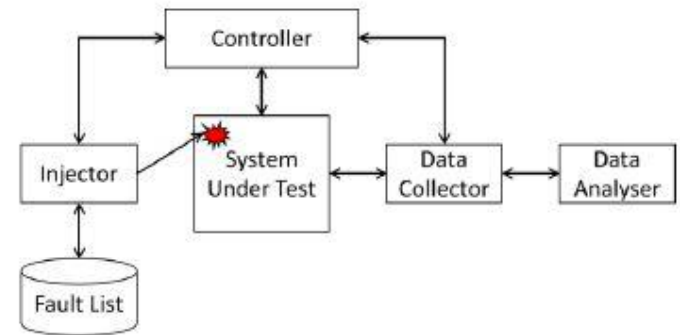
Layout Fault Model

- Layout fault model predicts the effects of laser attacks using layout information
- Three step procedure:
 1. A spot partitioning of the layout is required
 - Each spot represents a possible laser attack
 2. The corresponding affected gate level instances per spot are obtained
 3. The spot is characterized either as detected or undetected

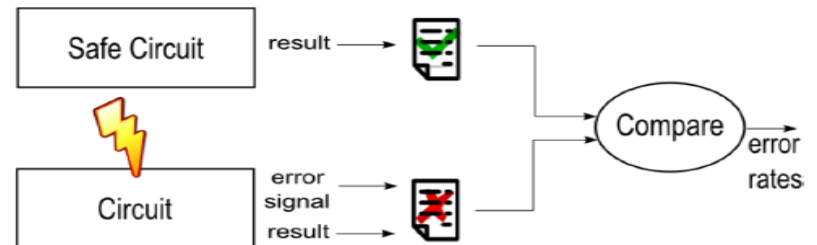


Fault injection campaigns

- Strategy:
 - the fault list indicates the cells to be “attacked”
 - we consider only flip-flops
 - inject faults and observe the effects



- Fault analysis:
 1. obtain the correct result
 2. obtain the result for each fault injection
 3. generate error rates



Countermeasures

- Countermeasures at RTL are based on redundancy

- Hardware redundancy

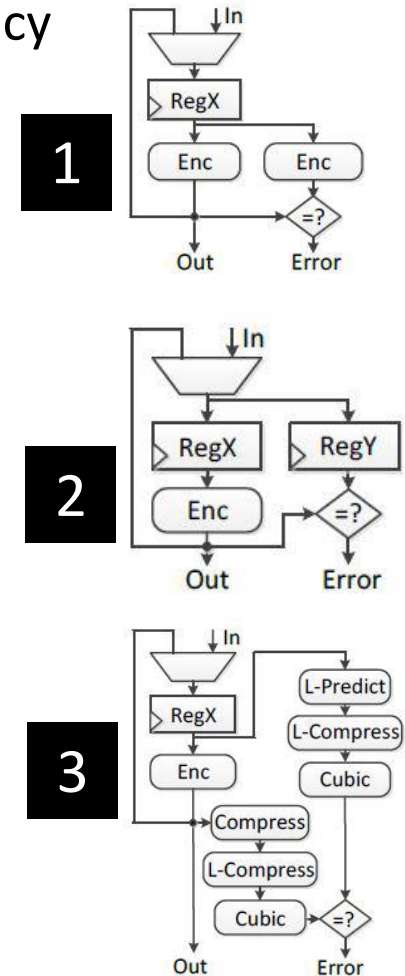
- Entails big area overhead

- Temporal redundancy

- Entails big performance overhead

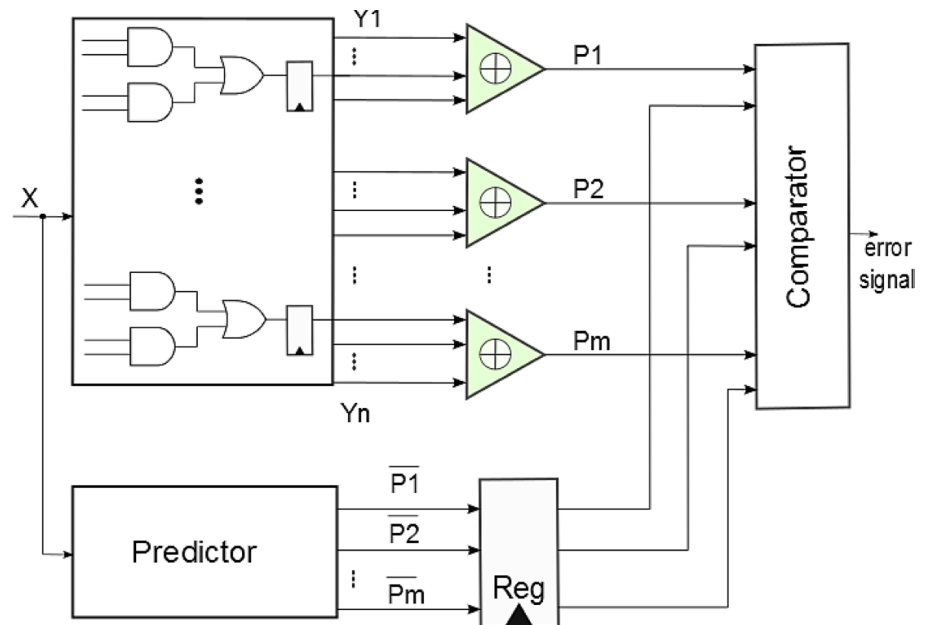
- Information redundancy

- Linear codes (parity)
 - + Cheap and simple
 - Do not detect even number of faults
- Non-linear codes (CRC, cubic)
 - + Detect all faults
- Big area overhead



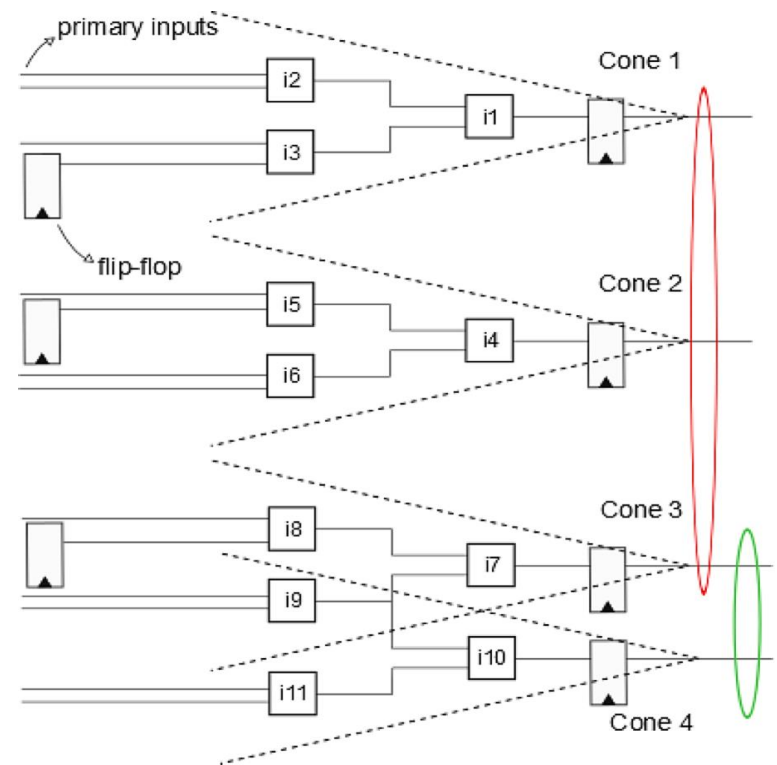
Countermeasure Architecture

- Architecture based on a hybrid approach
 - Mixing spatial hardware (duplication) and information (parity) redundancy
- Three procedural steps:
 1. Parity calculation from the original Design
 2. Parity prediction from the duplicated components
 3. Parities comparison
- Parity groups are the key factor
 - One fault can be found, at most, on each group



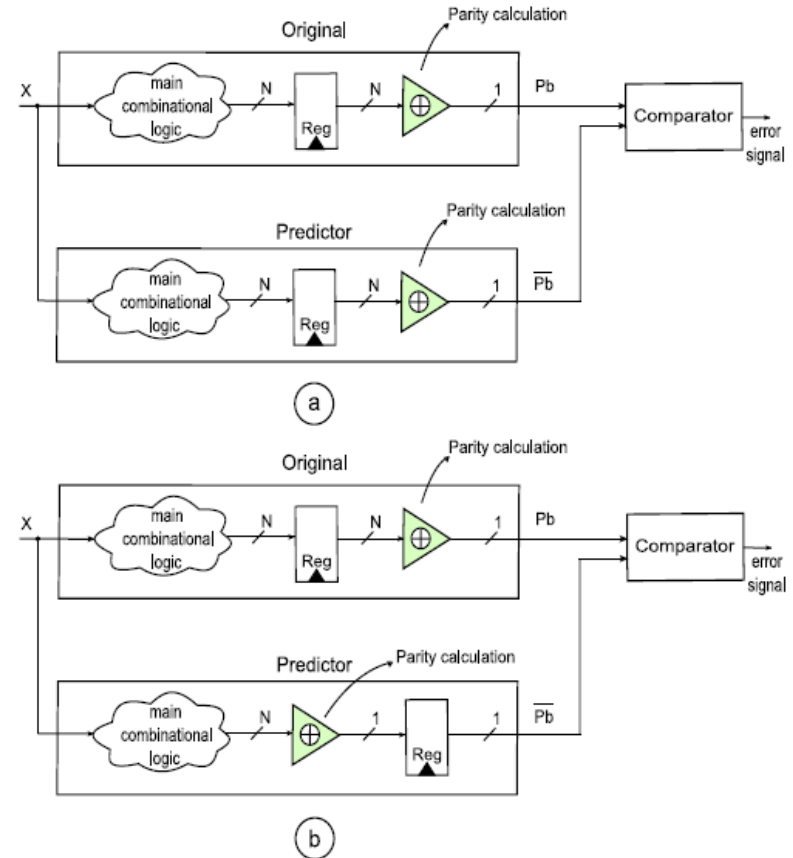
Construction of parity groups

- Three factors affecting the size and number of parity groups:
 1. Laser attack RTL fault model
 - Independent cones are the candidates
 - A laser attack is more likely to affect dependent cone than independent cones after layout
 2. Robustness vs Area
 - The possibility of an even fault to be injected increases as the parity group gets bigger
 3. Architectural characteristics of the circuit in conjunction with LAFM
 - Combine all the candidate bits in parity groups

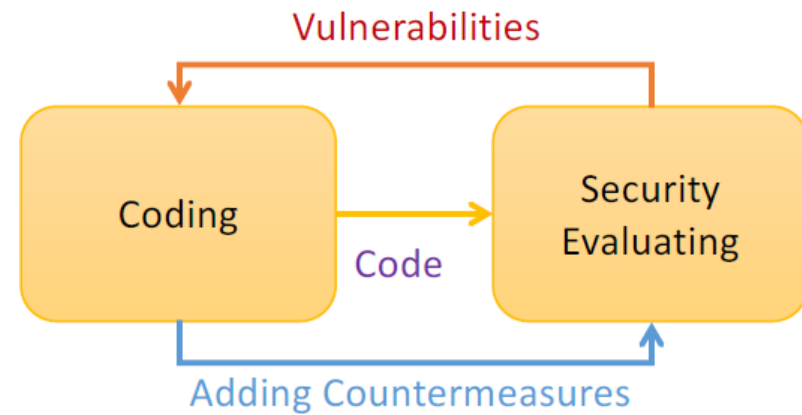
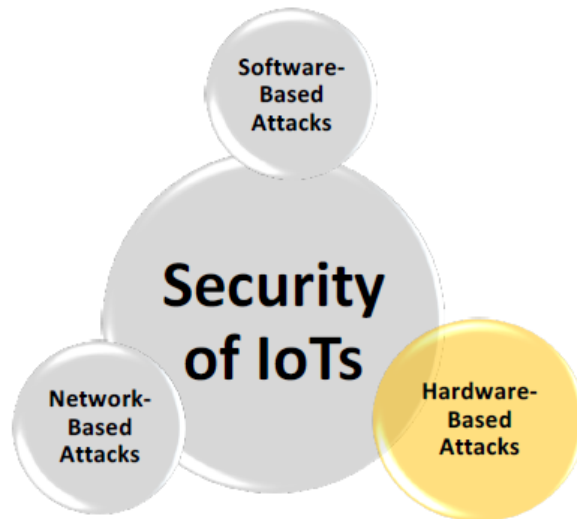


Area overhead reduction

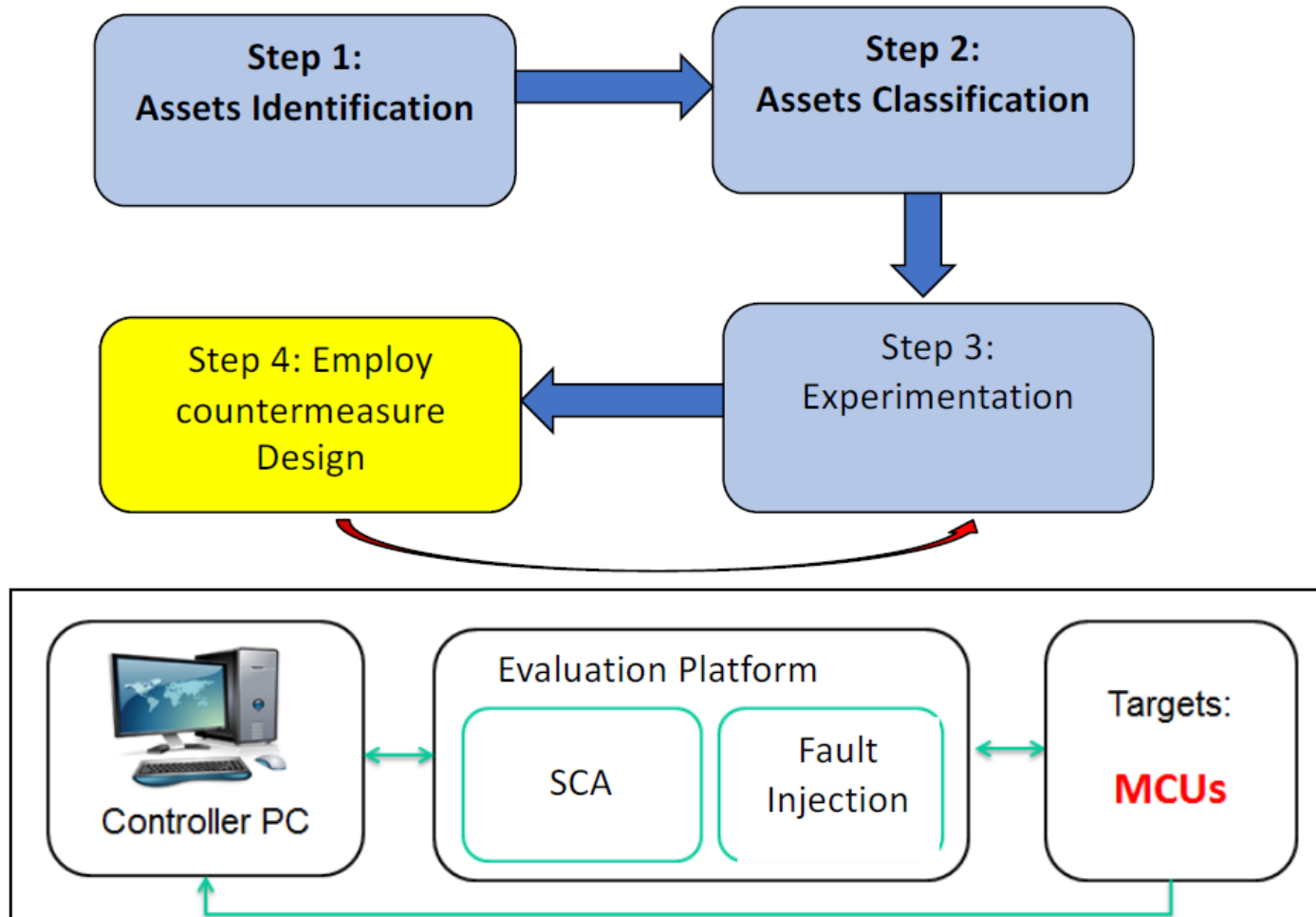
- Two strategies to reduce the overhead impact:
 1. Optimize the design with the EDA tool compilation options
 2. Change the way parity is calculated in the predictor
 - registers in the predictor are smaller
 - approximately 80% reduction in the utilized flip-flops



SW Evaluation

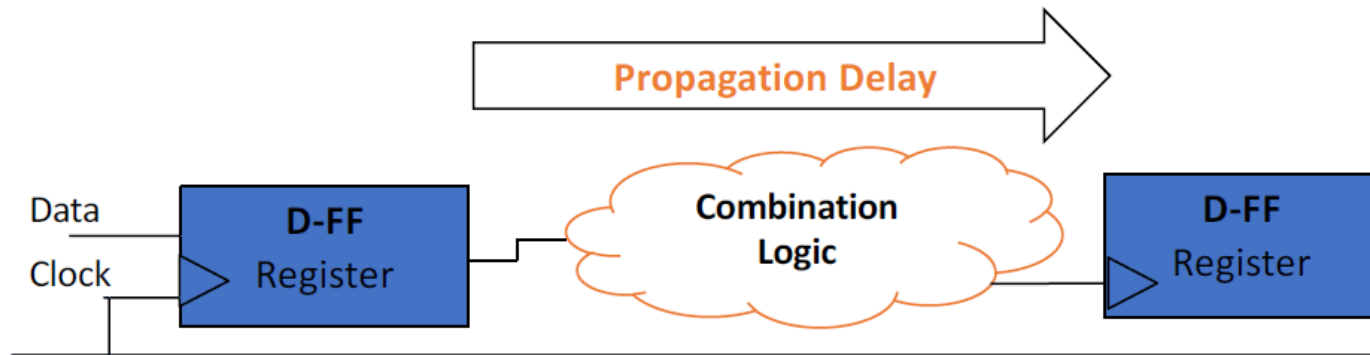


Experimental FI Evaluation



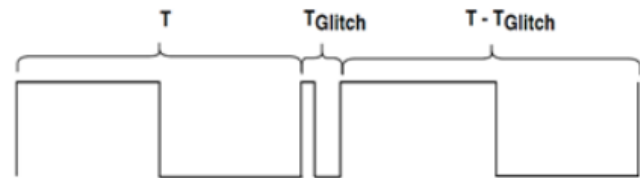
Clock glitching against embedded SW

- Proper Timing and Synchronous Representation of Digital ICs:



- Normal functioning timing rules:

$$T_{\text{glitch}} < T_{\text{propag. delay}} + T_{\text{set-up}}$$

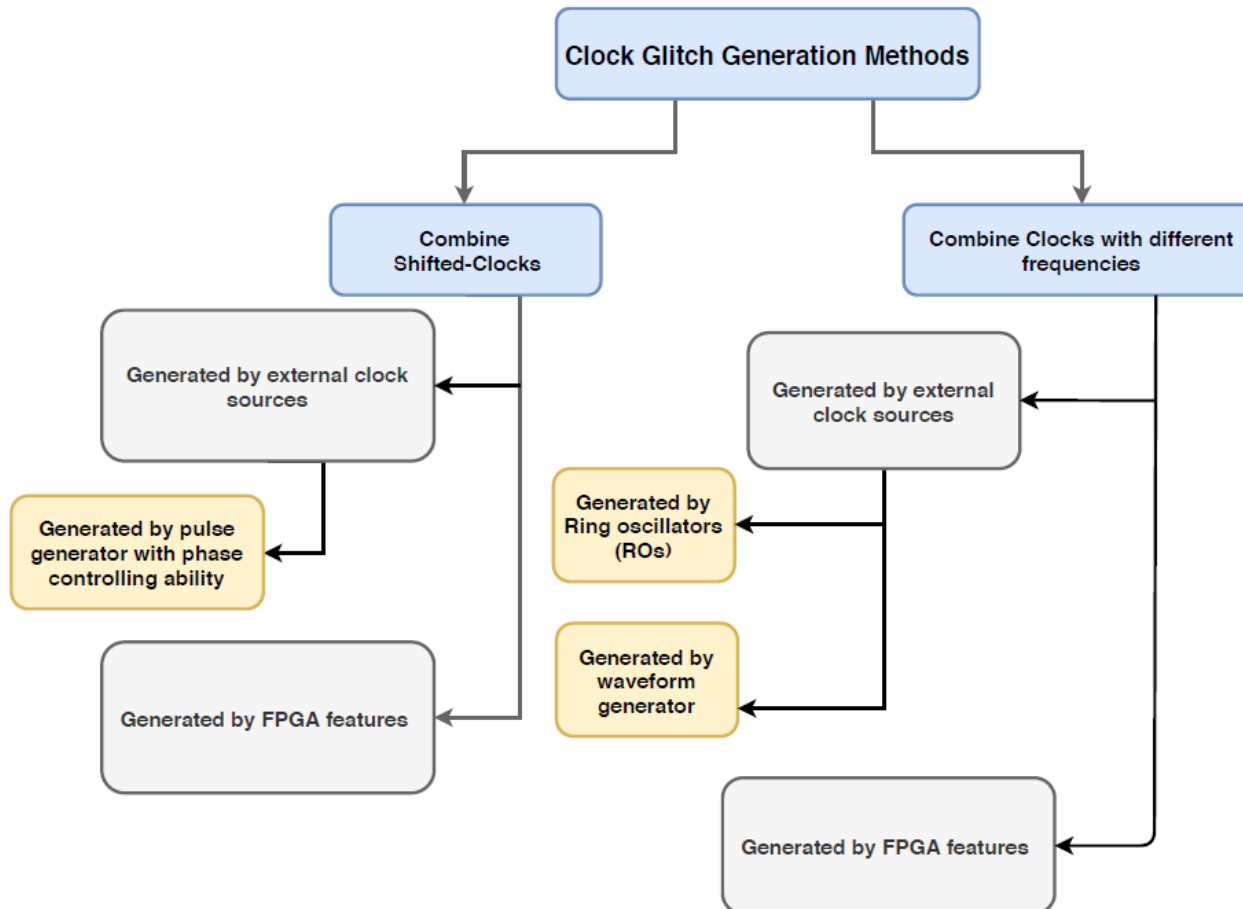


Clock Glitching vs Over Clocking

Technique	Spatial precision	Temporal precision	Controlling the Intensity	Equipment Cost	Needed Expertise
Overclocking	Low (global)	None (global)	Clock frequency	Low	Low
Clock glitch	Low (global)	High (local)	Glitch width and injection location	Low	Moderate

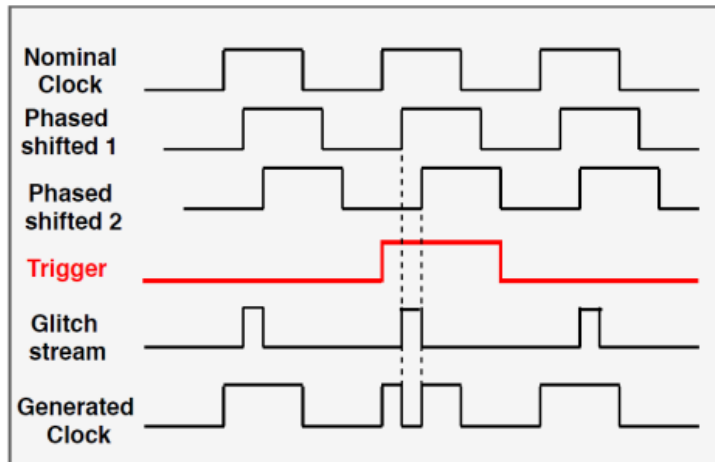
Clock Glitch Implementation

- There are two main categories to generate the glitch:

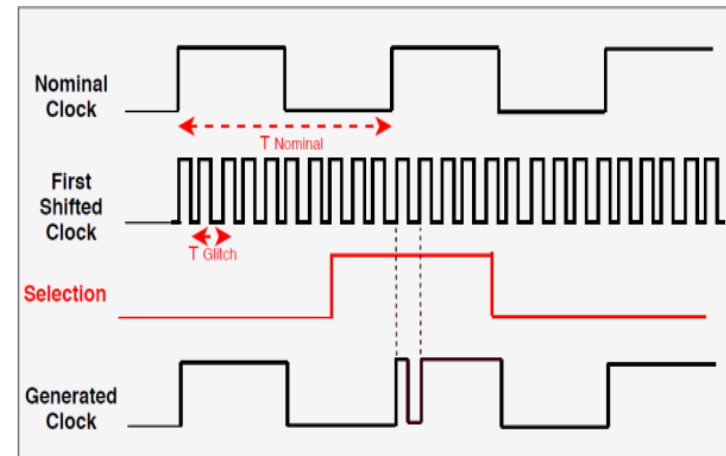


Clock Glitch Implementation

- There are two main categories to generate the glitch:



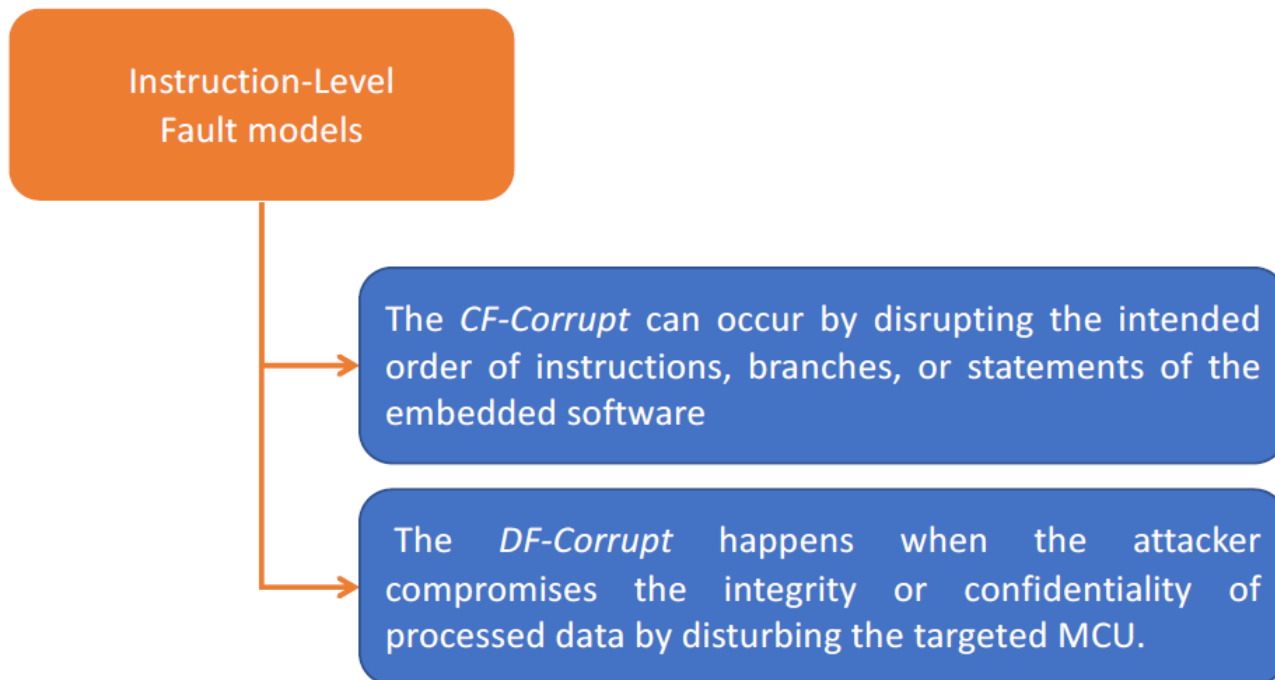
(1) Combine Shifted Clocks (CSC)



(2) Combine Clocks with Different Frequencies (CDCF)

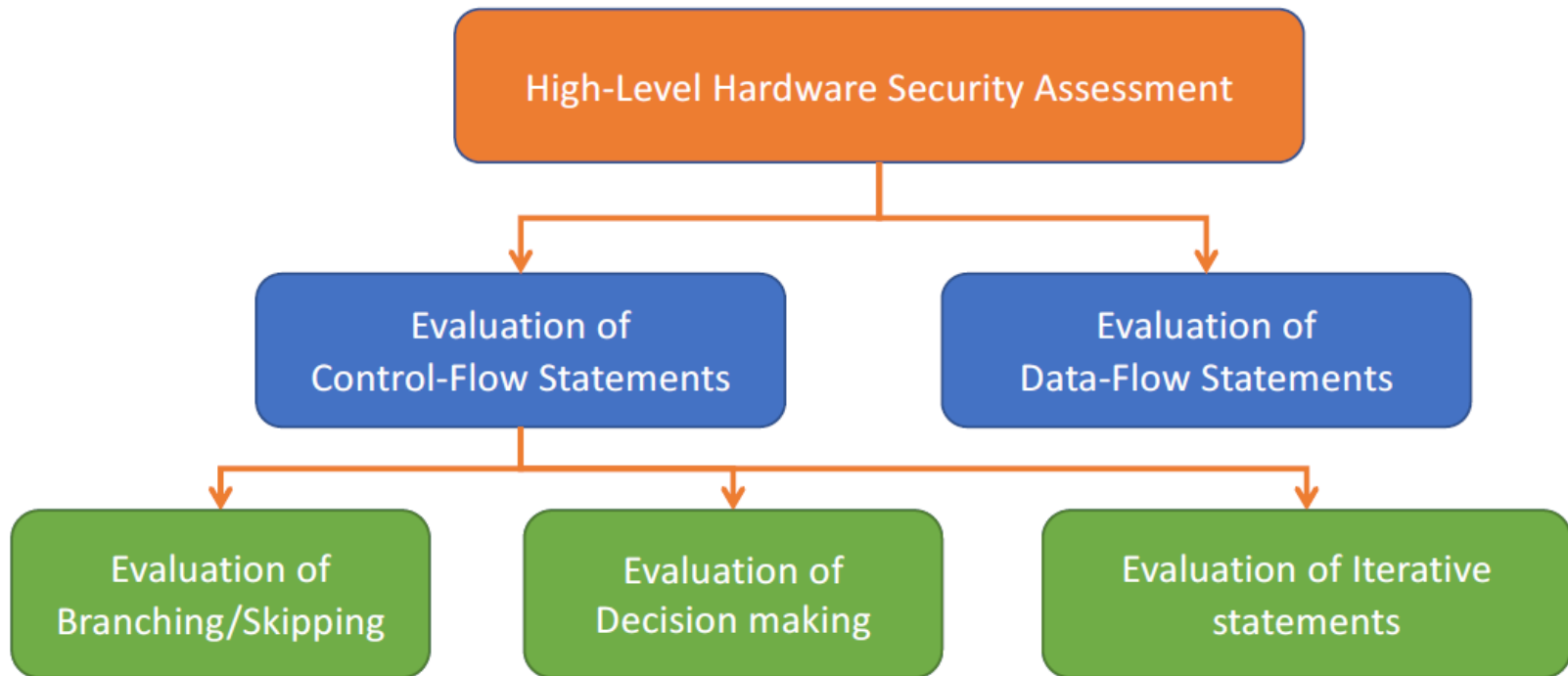
Impact of clock Glitching Attacks

- The clock glitching attacks can result in high-level (instruction-level) fault consequences or fault models:



Security Assessment

- Hardware security evaluation of the program in the presence of fault injection:



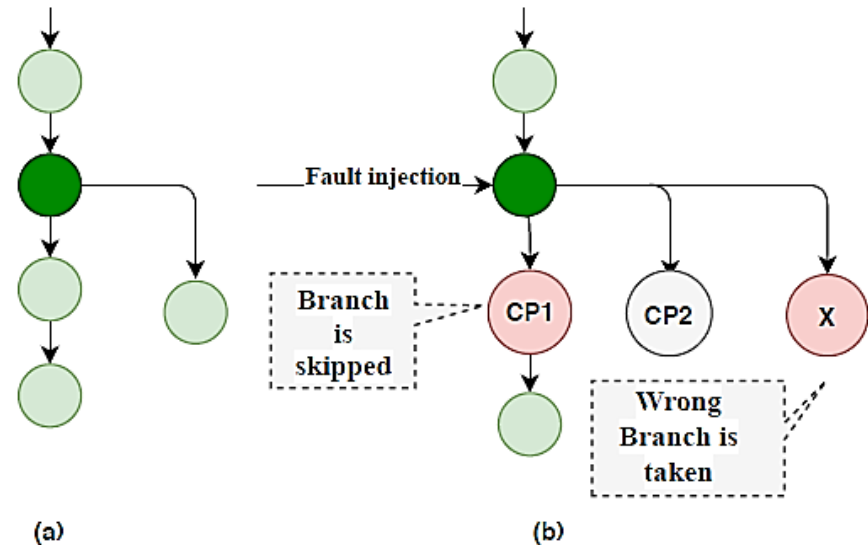
Security Assessment

- Evaluation of the Control-Flow Statements:

Control Flow Statements	Type	Examples
Branching/Skipping	Unconditional	Continue/Break/Go
Decision making	Conditional	If/ If-else
Iterative	Conditional	For/ Do-while /While

Evaluation of the Control-Flow Statements

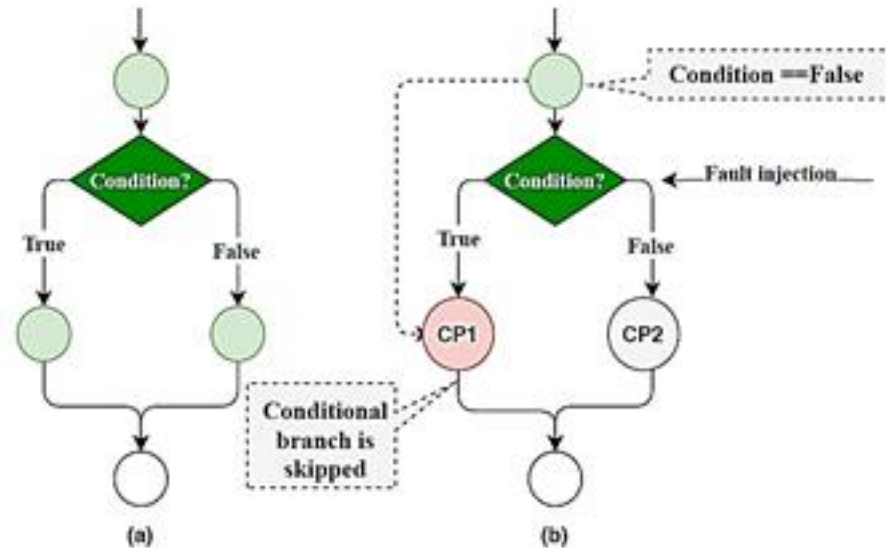
1. Unconditional Branch:



- We assume two checkpoints CP1 and CP2
- CP1 is activated → the injected fault did not affect the correct execution of the branch
- CP2 is set → the branch was corrupted
- When none of the checkpoints are activated → the PC register contains an incorrect instruction memory address (represented as X).

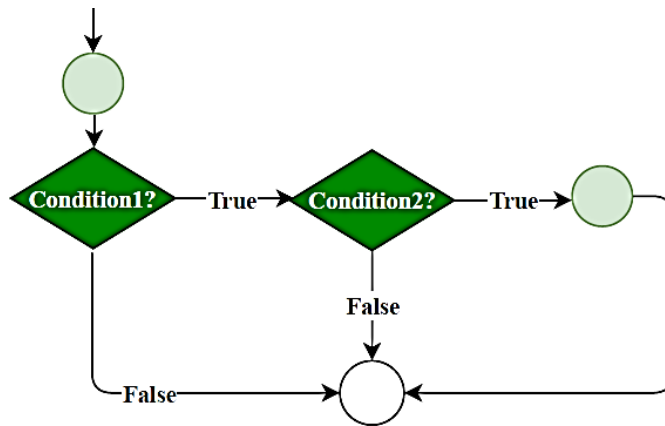
Evaluation of the Control-Flow Statements

2. Single Conditional Branch:

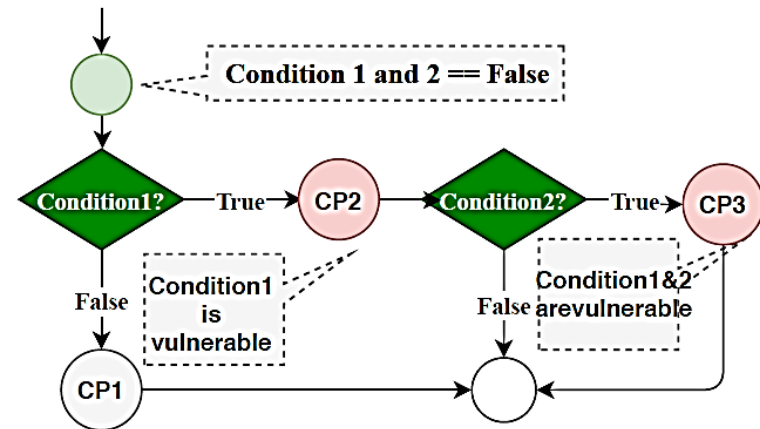


- We assume two checkpoints CP1 and CP2
- We expect the condition to be false
- CP2 is activated → when the fault injection is unsuccessful
- CP1 is set → skipped conditional test is skipped

3. Branch with nested conditions:



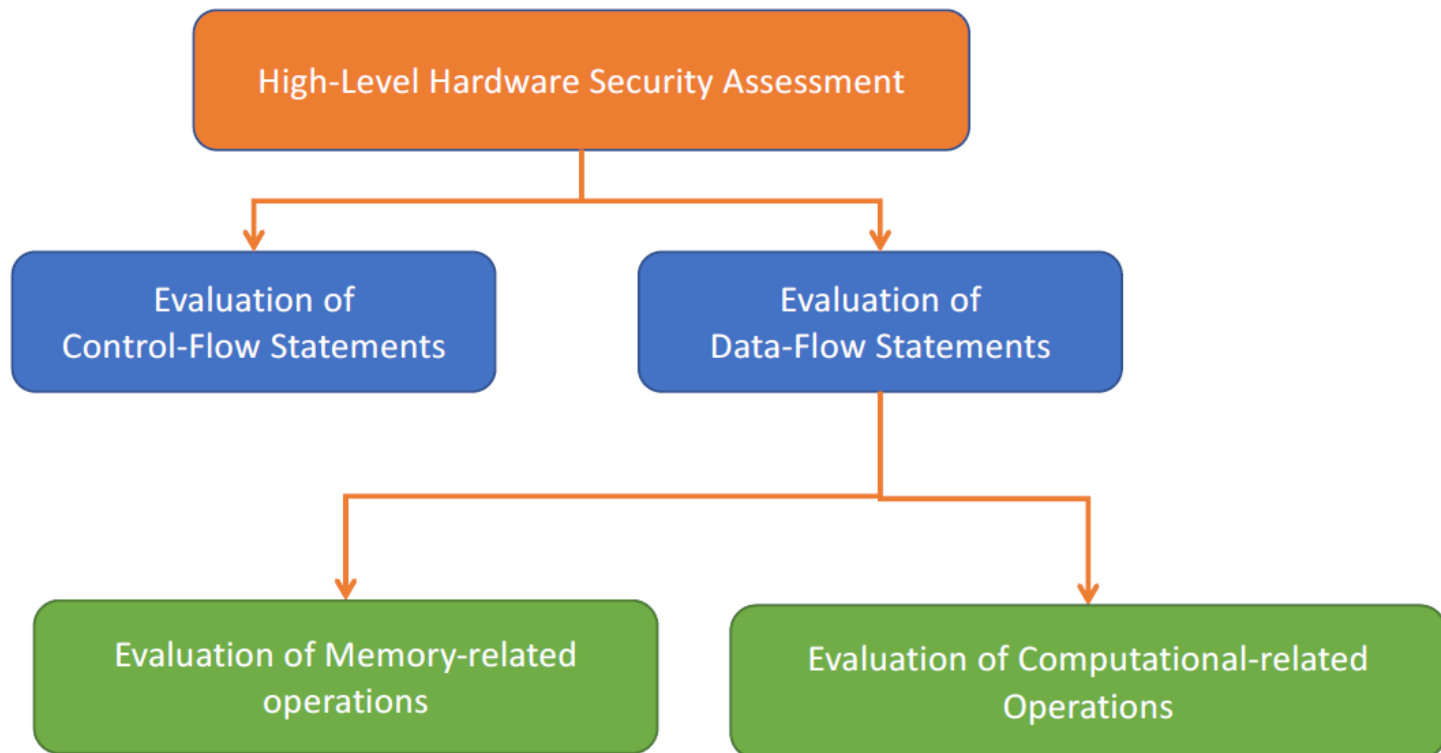
(a)



(b)

- We set the conditions to give false results
- We expect the CP3 to be activated
- The activation of CP1/CP2 and we can detect the vulnerability of branches.

Evaluation of the Data-Flow Statements



Evaluation of the Data-Flow Statements

- Data flow statements represent the pattern of data assignment and data usage through the program execution.
 - Evaluation of Memory-related operations
 - Evaluation of Computational-related Operations

Data Flow Statements	Type	Examples
Data assignments	Memory Access	Memset/Strncpy/Strncpy
Data usage	ALU operations	Add/Sub/Multiply/atoi

Important Data-Flow Statements

Fault Injection Evaluation?

- What do we want to evaluate:
 - A circuit?
 - i.e. a digital circuit designer wants to evaluate the vulnerabilities of the circuit against FI.
 - A program?
 - A software developer wants to evaluate how the program will behave in case of FI.
- When do we want to evaluate?
 - At design time?
 - How do we model the FI process?
 - On which system description do we perform the evaluation?
 - On the deployed system?
 - Can we emulate FI?
 - Can we use FI equipment?

Questions?

amir-pasha.mirbaha@lcis.grenoble-inp.fr