Fault Analysis in RTL microarchitectures and HW/SW countermeasures

Johan Laurent, **Vincent Beroulle,** Christophe Deleuze
Grenoble LCIS, Valence
Firstname.Lastname@lcis.grenoble-inp.fr
Florian Pebay-Peyroula
CEA-Leti, LSOSP
florian.pebay@cea.fr

Agenda

- **Introduction**
  - LCIS Lab/CTSYS team
  - Project foundations
  - Security evaluation platforms and problematics
  - Case study and goals
- **VerifyPin case study**
- **Conclusion**

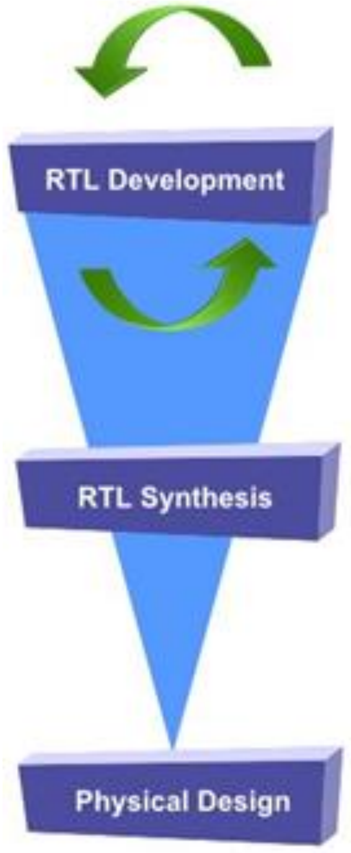
Introduction
LCIS/CTSYS team

- LCIS: **COMUE UGA lab located in Valence**
- CTSYS: **9 researchers on the « Security of embedded systems and distributed systems »**

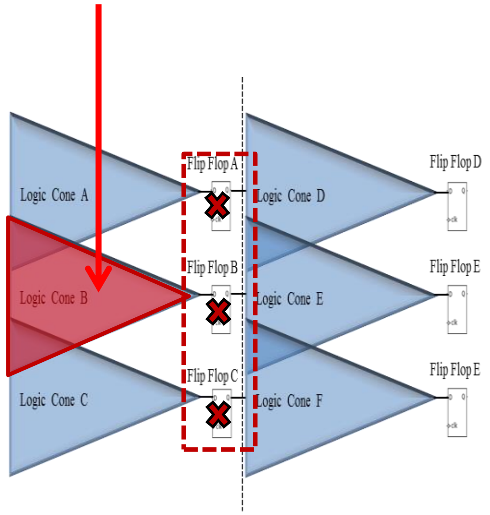
- **Interdisciplinarity**: taking into account interaction between hardware and software

CSAW'19 EUROPE
Grenoble INP - Esisar
6-8 NOVEMBER 2019
Grenoble INP Esisar

Introduction
Project foundations



Localized fault attacks (laser, EM, etc)



Validated with laser on 28 nm Bulk (STM) ANR LIESSE

$$\begin{tikzcd}
\boxed{\text{RTL FAULT MODEL}} \arrow[r] \arrow[d] & \text{SEU / MBU} \\
\boxed{\text{RTL FAULT INJECTION}} \arrow[r, phantom, ""] & \\
\end{tikzcd}$$

RTL FAULT MODEL → **SEU / MBU**

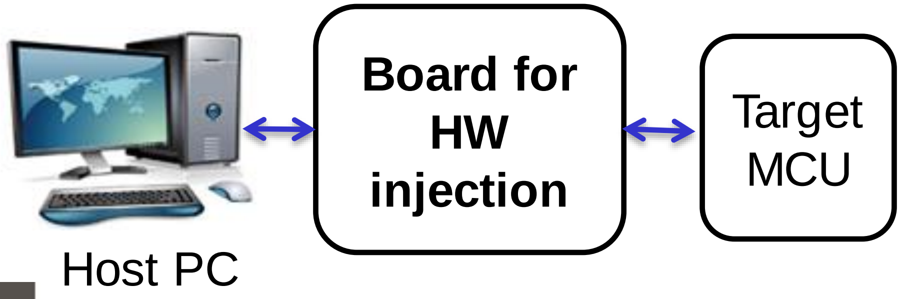
RTL FAULT INJECTION → **RTL CM DESIGN**

Automatic RTL Fault Model Extractor and Simulator
Automatic HW CM Generator

Introduction
Security evaluation platforms

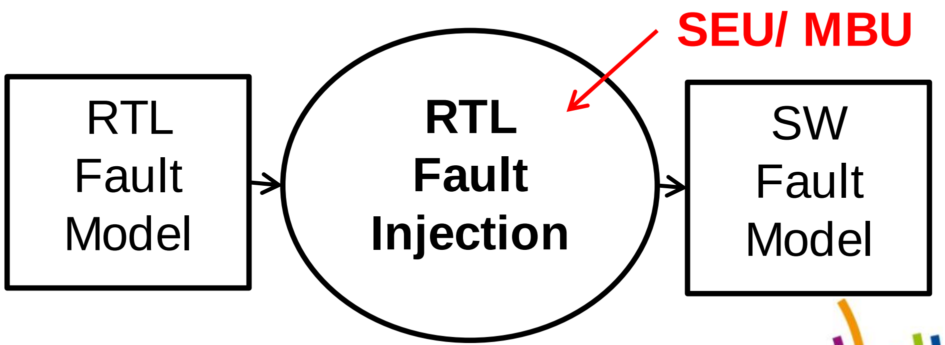
- **Embedded software developers need tools to:**
  - Analyze the hardware threats to demonstrate the vulnerabilities
  - Perform early evaluation of their designs and countermeasures
- **2 platforms: HW-based vs Sim-based Fault Injection**

HW injection for evaluation of the system



SereneIoT Project

Cross Layer fault model for **early security HW/SW co-simulation** of the system

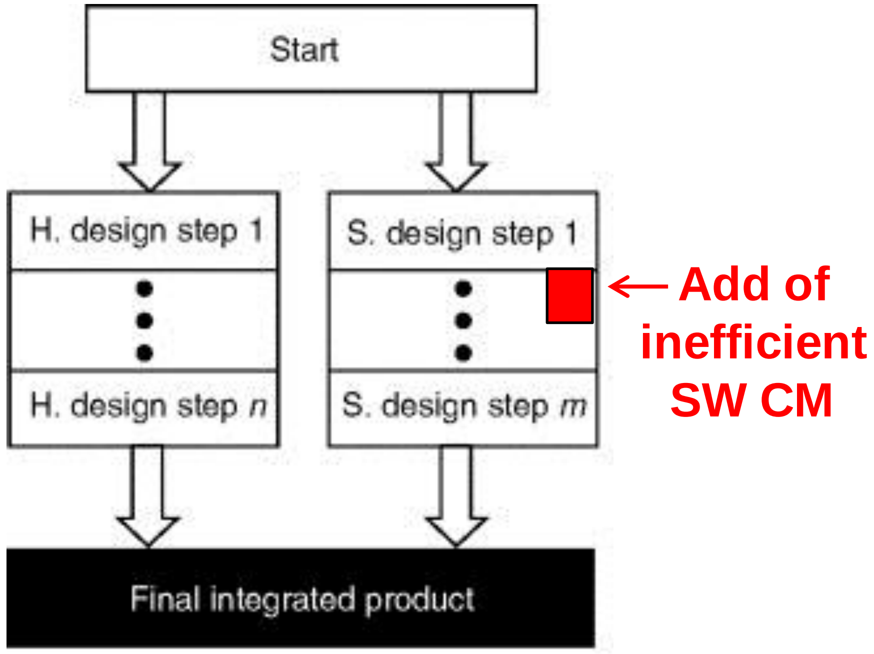


Secure-RTL Project

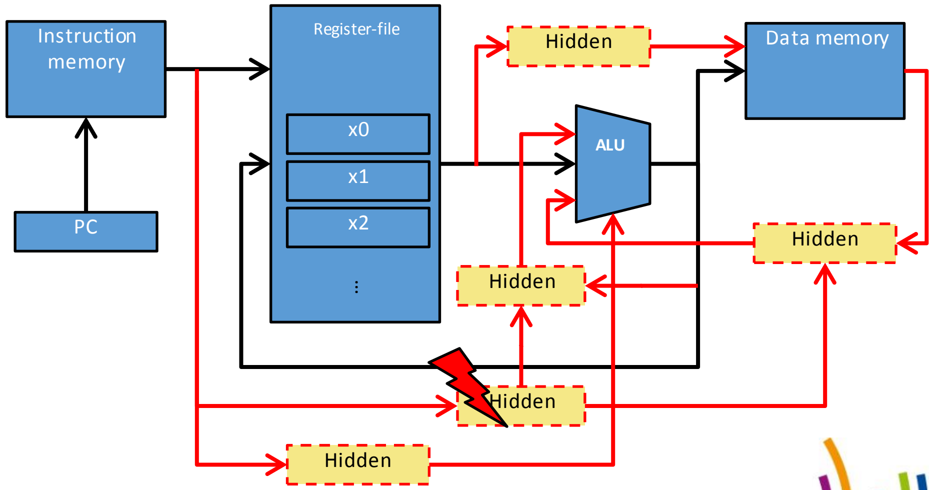
Introduction

Simulation-based security evaluation platform

- Typical Design Flow

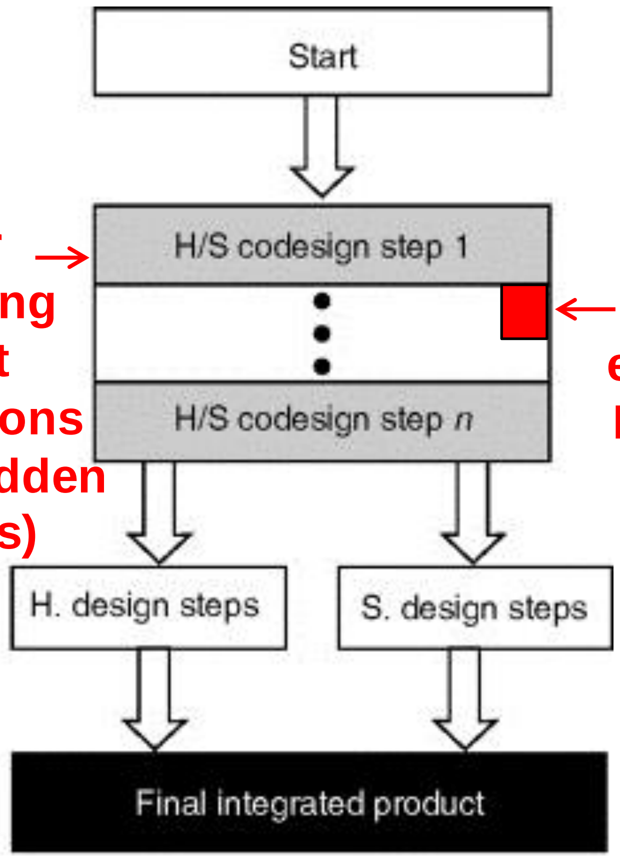


- Typical SW fault models do not take into account HW microarchitecture
- HW hidden register fault effects can bypass SW CM



Introduction
Simulation-based security evaluation platform

- **HW/SW Co-design Flow** (RISC-V opportunities)



Easier monitoring of fault propagations (even in hidden registers)

Add of efficient HW/SW CM

- **Analysis of HW RTL microarchitecture: new SW Fault Models**

⇩

- **SW Fault injection for detecting security breaches**

⇩

- **New SW (or HW) CM to prevent security breaches**

Introduction

Case study context and goals

- **Case study on a secure code :** VerifyPIN
  - from FISCC (Fault injection and Simulation Secure Code Collection) proposed by Verimag
  - with HW fault simulation on RISC-V Rocket processor (RTL)
- **Goals:**
  - To highlight the importance of hidden registers in the processor pipeline
  - New SW CM proposals

Agenda

- **Introduction**
- **VerifyPin case study**
  - VerifyPin SW CM and description
  - RISC-V Rocket : forwarding detection
  - Cross Layer SW fault model extraction
  - New fault attacks and SW CM
- **Conclusion**

Case study
VerifyPIN

- VerifyPIN: **simple code comparing 4-digit PIN values**
- **8 versions of SW CM:**
  - Hardened Booleans
  - Check loop counter at the end
  - Double boolean tests
  - Inlined calls
  - Step counter

Case study
VerifyPIN

```
diff=FALSE; status=FALSE; //hardened booleans

for(i=0 ; i<4 ; i++){
        if(userPIN[i]!=cardPIN[i])
                diff=TRUE;
}

if(i != 4) countermeasure(); //check loop counter

if(diff==FALSE)
        if(FALSE==diff)    //double tests
                status=TRUE;
        else
                countermeasure();
else status=FALSE;

return status;
```

Pseudo-code of the application

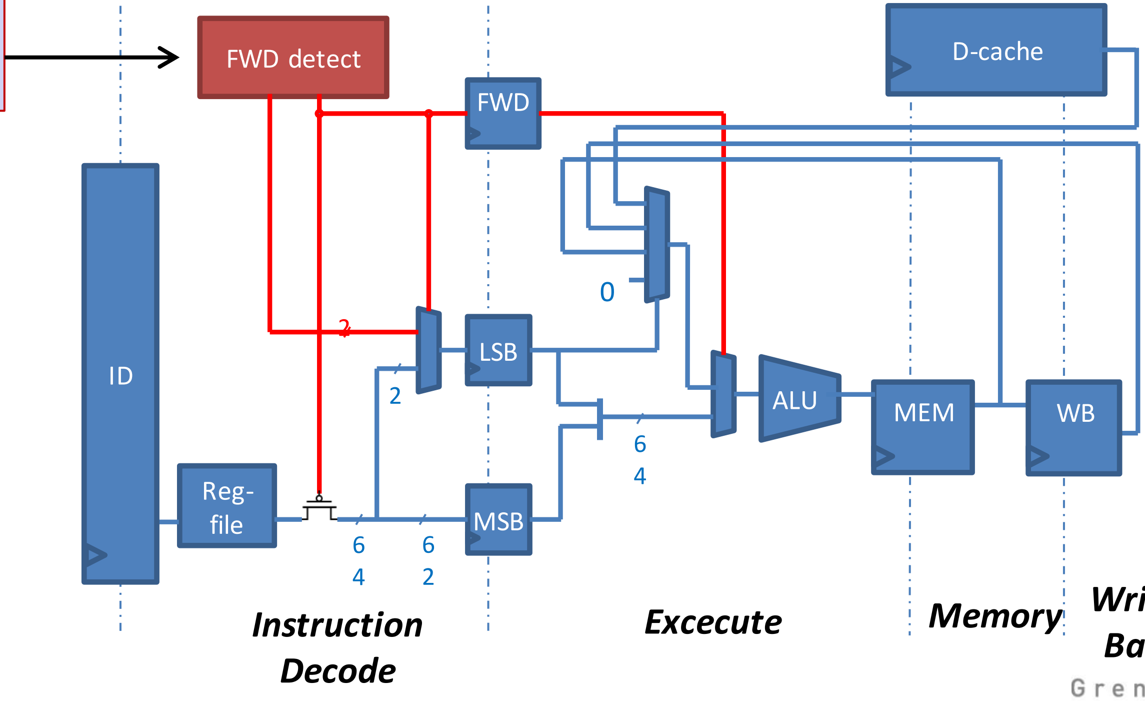
Case study RISC-V

- **RISC-V open HW processor architecture:** LowRISC v0.2., 64-bit Rocket core implementation

ADD	x3 = x1 + x2
ADD	x4 = x3 + x2

Example of a data hazard

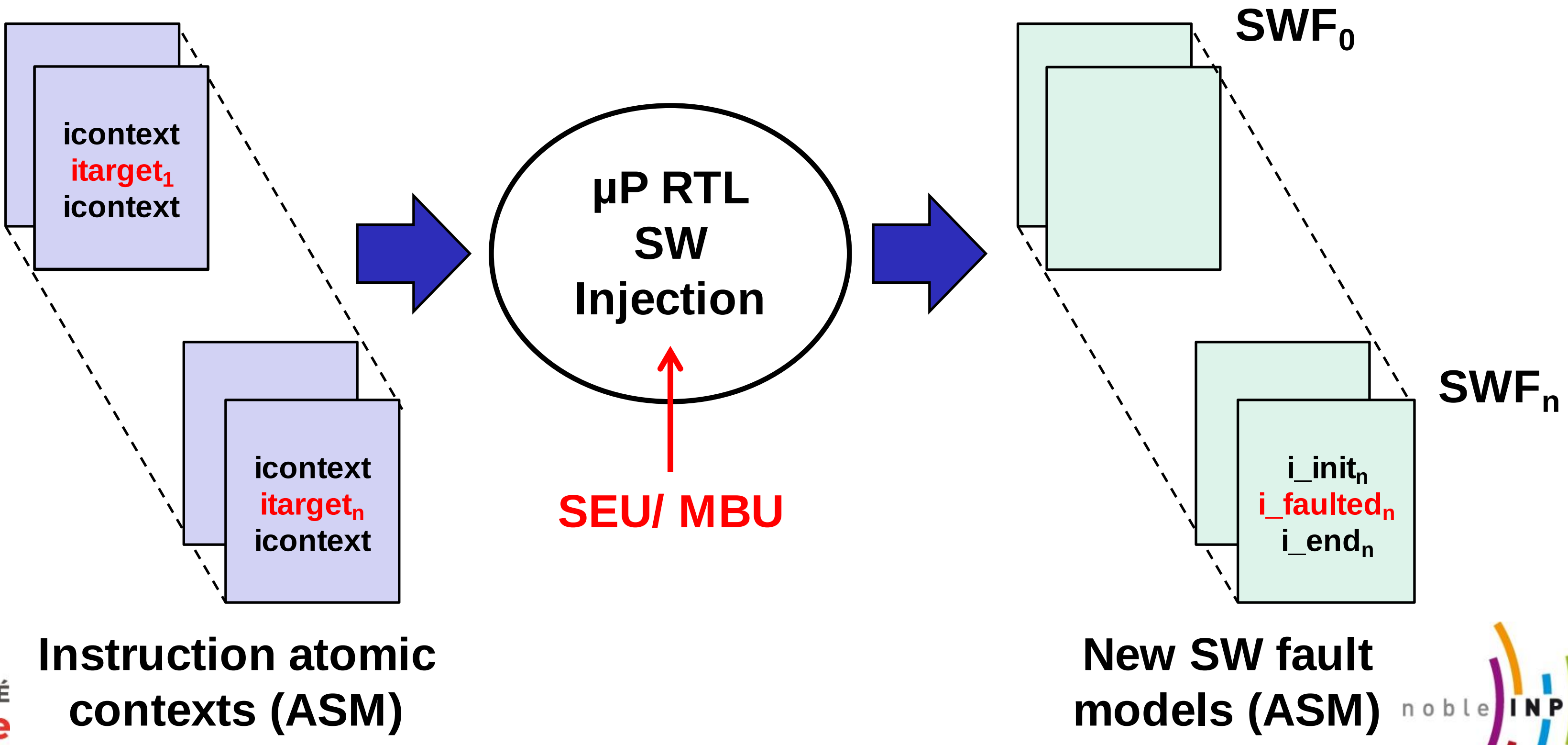
Forwarding detection



Cross Layer SW Model Extraction

SW Faulty Behavior Characterization

- **Analysis of HW RTL microarchitecture:** new SW Fault Models



Case study
SW fault models characterization

Instruction	Origin	
Branch	Branch	
	Mux_1 or Mux_2	
	ALU_op	…
	Write_enable	…
	(not represented)	…
R-type	Write_enable	SWF1
	Branch	…
	Mux_1 or mux_2	
	ALU_op	…
Load	Write_enable	…
	Ctrl_mem	
	ALU_op	
	Mem_cmd	
	Mem_cmd	
	Mem_cmd	
Store	Ctrl_mem	
	ALU_op	
	Write_enable	
	En_store	
	Mem_cmd	
Jump (jal)	Write_enable	
	Mux_2	
	Jal	
	(not represented)	…

Here faults are injected in control signals only

New SW faulty behaviors are characterized

SWF1: new SW fault model

```
ADD    x3 = x1 + x2
ADD    x4 = x3 + x2
```

$\Rightarrow$

```
// skip
ADD x4 = x1 + x2 + x2
```

Due to forwarding, x4 is fault free but x3 does not store x1+x2

Case study
New fault attacks

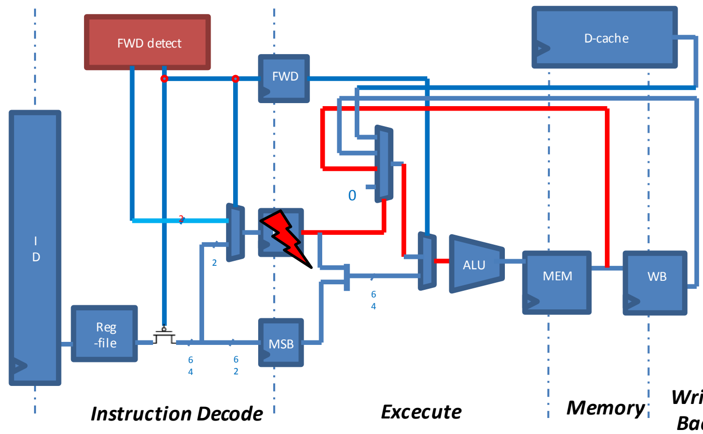
- **Hardened boolean** : to be safe against single bit fault injection (*false=0x55, true=0xAA*)

ADDI a5 = 0 + 0x55
…
ADDI a5 = 0 + 0x55

Forwarding of the 0 value

Single bit fault injection during forwarding

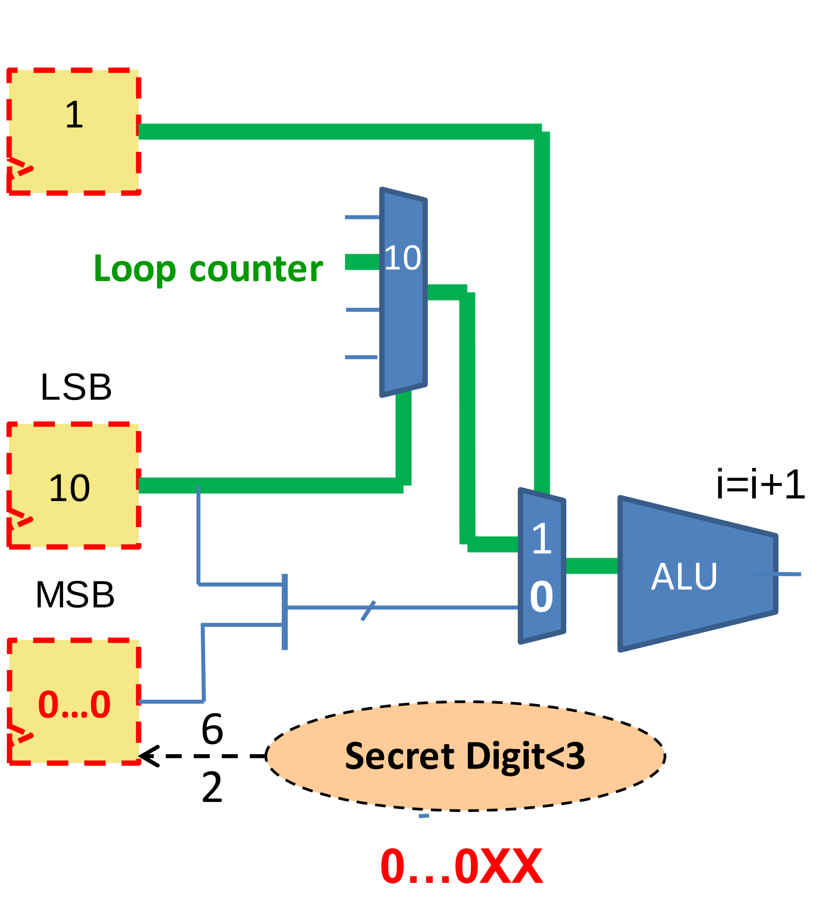
ADDI a5 = 0 + 0x55
…
ADDI a5 = 0x55 + 0x55 = 0xAA!!!

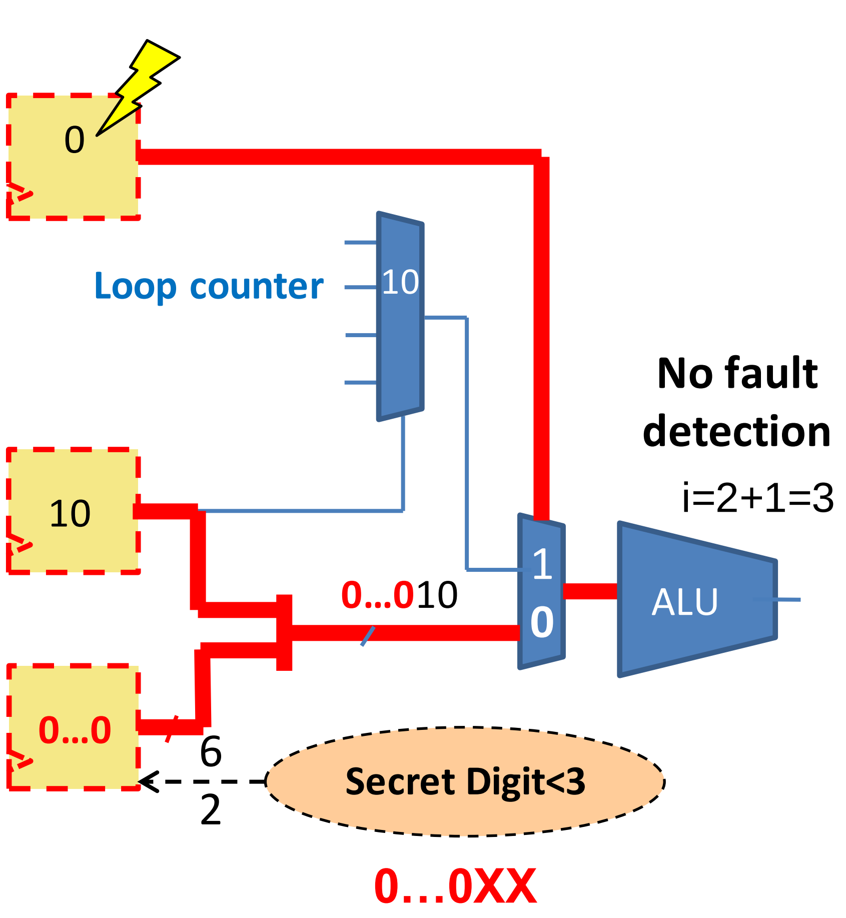


Case study
New fault attacks

- **Context: A countermeasure checks if the loop was executed 4 times**
- **Goal: Safe-error attacks**
  - Thanks to fault injection, make the CM trigger or not <u>depending of the value of the Secret Digit</u>
- **How to do it:**
  - Force the use of the Secret Digit instead of the loop counter in the loop comparison

Case study
New fault attacks





How to protect: Simply swap arguments!

if(userPIN[i] != cardPIN[i]) ➔ *if(cardPIN[i] != userPIN[i])*

Conclusion

- **Hidden registers in complex RTL microarchitectures can generate complex faulty behaviors**
- **Complex faulty behaviors create vulnerabilities impossible to manage with typical SW CM only**
- **Cross layer analysis of the RTL microarchitecture is a required step to design effective HW/SW CM**

- **Perspectives:** Automate the vulnerability analyze for a given application and a given processor architecture

Project funding

- Thank you for your attention!

- This work was funded thanks to the French national program 'programme d’Investissements d’Avenir, IRT Nanoelec' ANR-10-AIRT-05