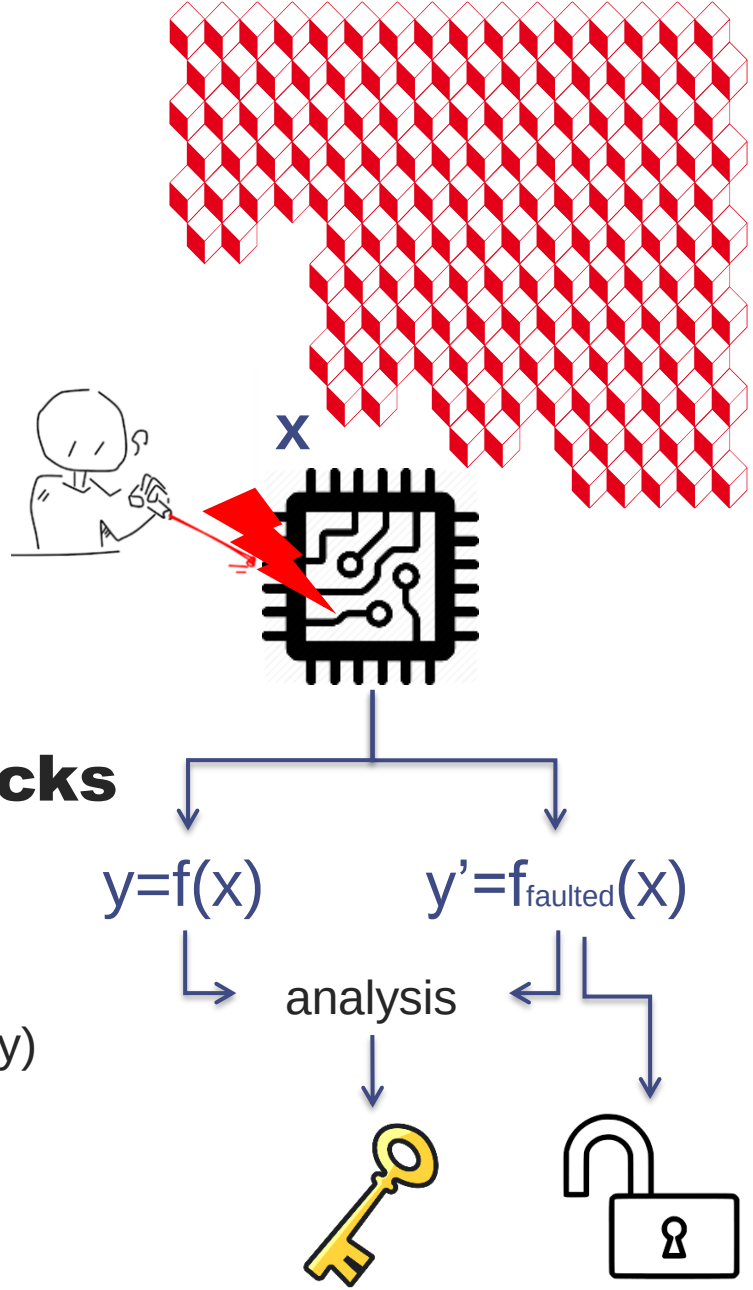


Pre-Silicon Analysis Against Fault Injection Attacks

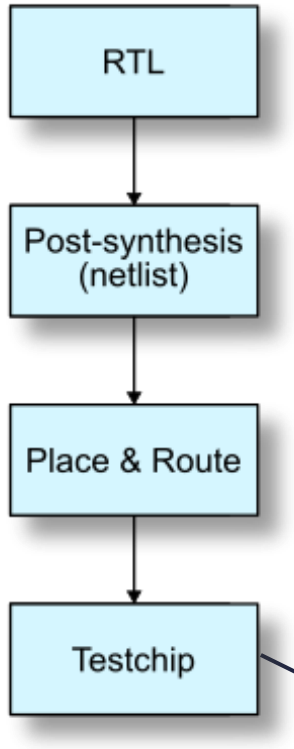
Jonah Alle Monne (CEA-List / Grenoble), Damien Couroussé (CEA-List / Grenoble), Giorgio Fardo (CEA-List / Grenoble), Mathieu Jan (CEA-List / Saclay)

March 2026

Context: secure element & fault analysis

What it is? How designed & evaluated?

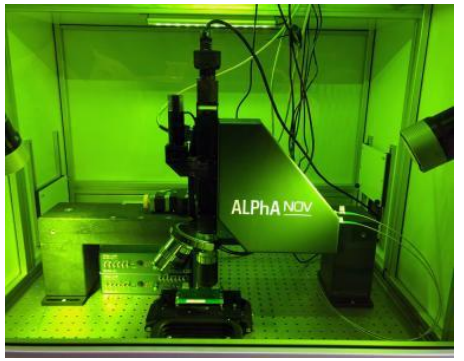
- Dedicated hardware component designed to protect assets (cryptographic keys) from unauthorized access
  - Within embedded (industrial) IoT devices, smartphones, smart cards, etc.
- Security evaluation: experimental pentesting using fault-injection and side-channels attacks performed by ITSEF labs



Can we anticipate through pre-silicon analyses?

- Late discovery of vulnerabilities: HW respins or additional software protections
- Automated Joint HW-SW analysis is mandatory!
  - Exhaustive analysis to provide guarantees → formal methods
  - Either SW or HW (accelerators) but none SW+HW and all using simulation

Photo from IMT/MSE

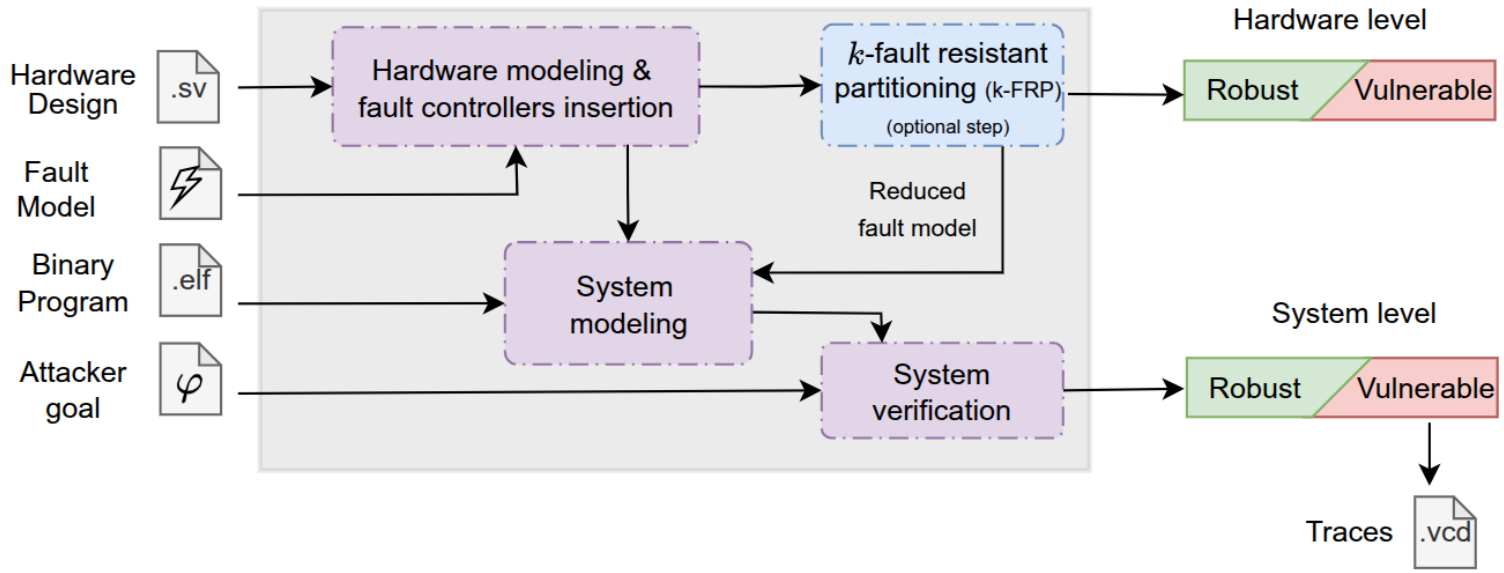


SECURE-IC THE SECURITY SCIENCE COMPANY CASPIA TECHNOLOGIES FortifyIQ OPTIMA

µArchiFI: our fault injection analysis tool

To do what?

- Identify fault effects at the HW level and check whether it generates vulnerabilities at SW level
- Prove the robustness of (HW/SW) countermeasures to secure a system
  - Leveraging hardware countermeasures to speedup analyses (k-FRP plugin)
- Reduce design costs and delays (avoid HW respin)



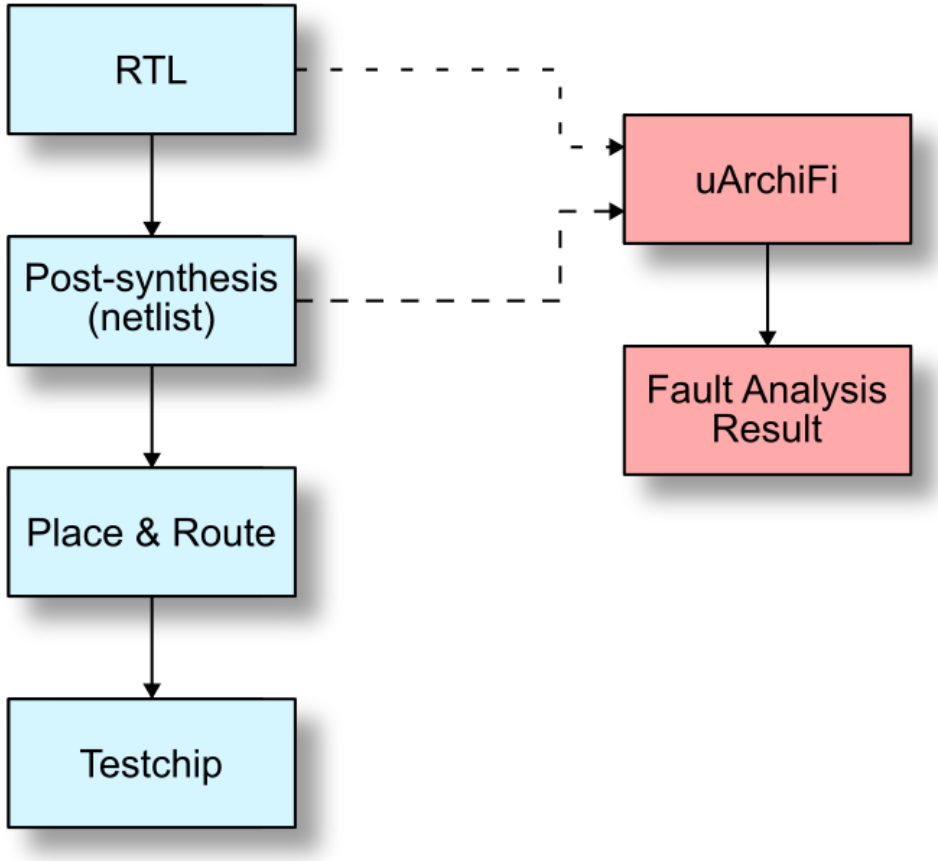
Tested platforms on µArchiFI

- HW: OpenHW Group processors (CV32P, CV32S), OpenTitan secure element (Secure Ibex), Crypto circuits (AES-128, etc.),
- SW level: FISCC benchmark, Tiny AES, SecureBoot

Now focusing on µArchiFI without k-FRP

How?

- Exhaustively analyze the impact of fault injections over systems (HW/SW) using formal methods
  - Configurable fault model: (gate) location, effect, timing and fault order
  - Analyses at RTL or netlist levels and agnostic to EDA flows
- Open-source flow: https://github.com/CEA-LIST/uArchiFI
- Container based environment for maximum portability



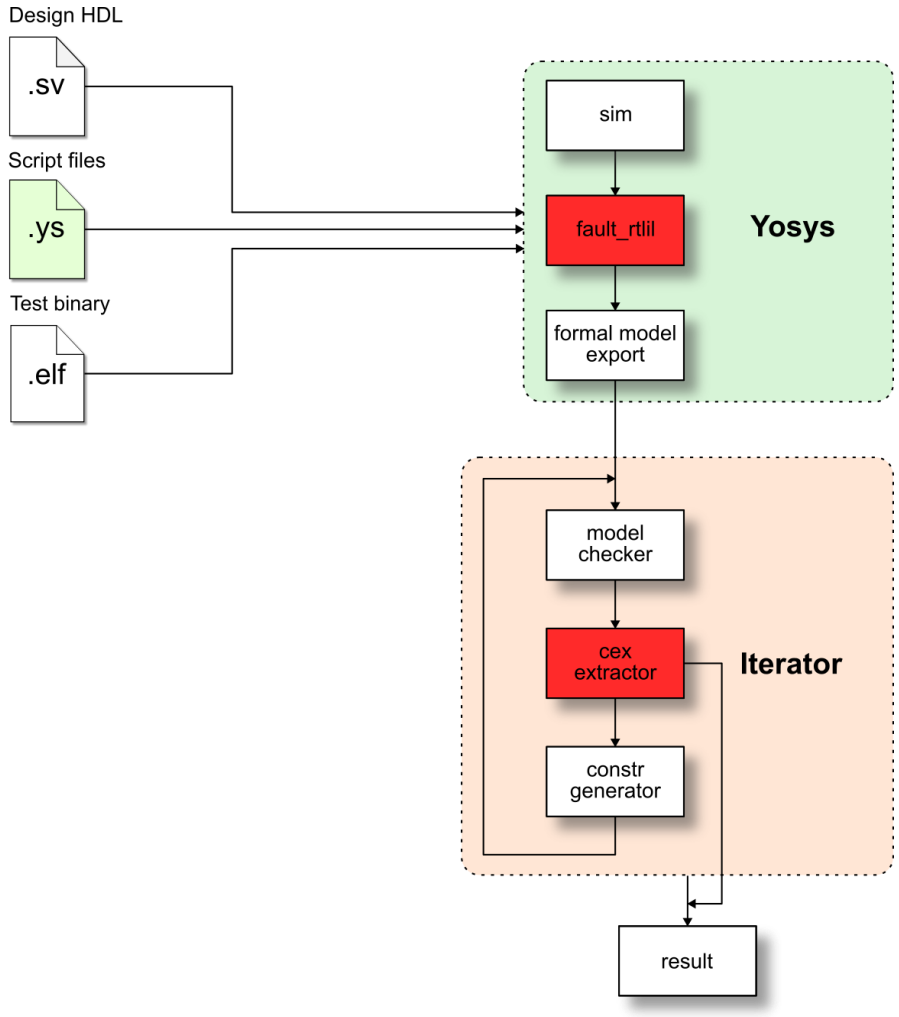
µArchiFI: internal architecture

Fully templated pipeline

- Simulation to reach desired state before formal checking
- Insertion of fault-injection multiplexers on selected signals
- Generation of formal models using Yosys synthesis tool

Multiple faulty traces iterator

- Leverages off-the-shelf model checkers (SMT-LIB/BTOR formats)
- VCD parser + constraint generator: extract and exclude faulted signals for next verifications
- Stop condition: #iterations and time bounds
- Produces evaluation report



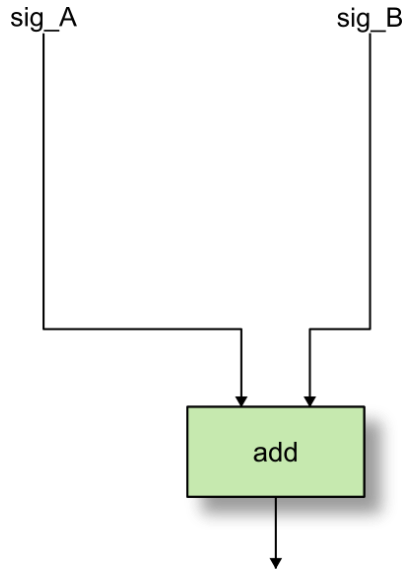
µArchiFI: fault_rtlil pass

Saboteurs injection

- Adds MUXes on the selected wires of the design
- Adds global fault counters and fault timing controllers
- Adds fault_sel signals that activate the saboteurs

Fault models

- Word-level fault model (i.e. modifying all bits): set, reset, flip, specific value and symbolic (all possible values)
- Recently added bit-flip level granularity (but mono-bit only)



Design before fault_rtlil pass

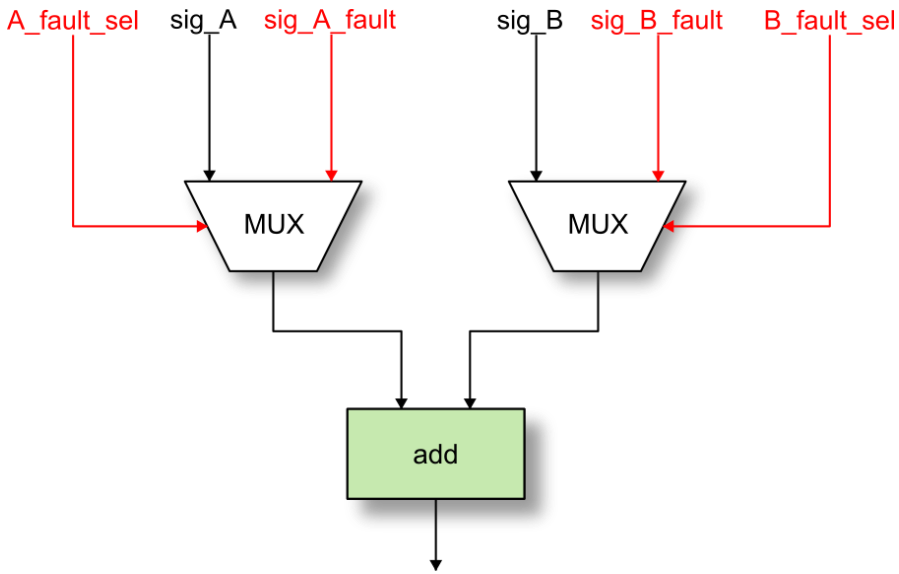
µArchiFI: fault_rtlil pass

Saboteurs injection

- Adds MUXes on the selected wires of the design
- Adds global fault counters and fault timing controllers
- Adds fault_sel signals that activate the saboteurs

Fault models

- Word-level fault model (i.e. modifying all bits): set, reset, flip, specific value and symbolic (all possible values)
- Recently added bit-flip level granularity (but mono-bit only)



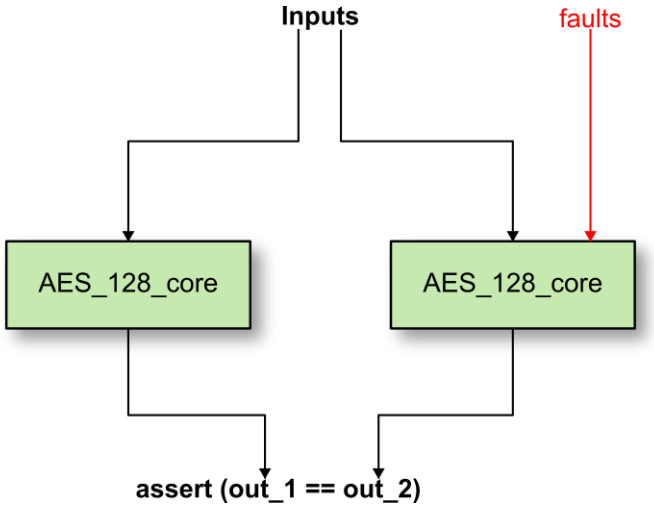
Design **after** fault_rtlil pass

Demo structure

Showcase of flow for Advanced Encryption Standard (AES) core implementation

(case study with only hardware)

1. Self-composition of the design
2. Scripts structure
3. Flow launch and Iterator running
4. Report showoff

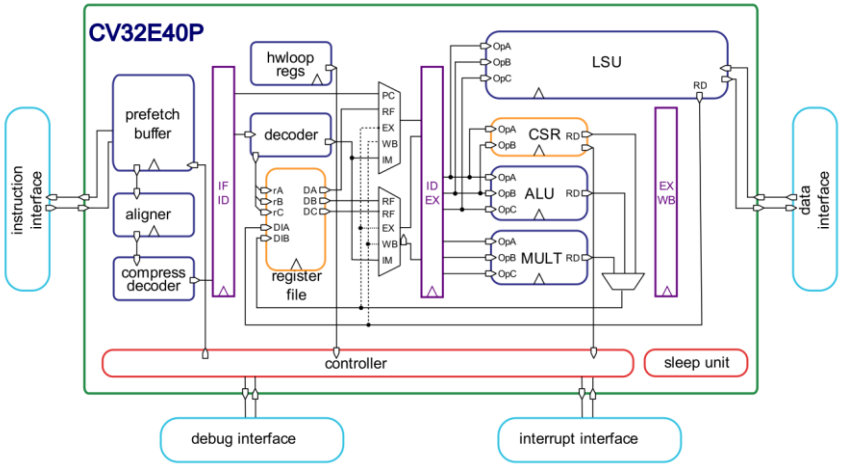


Self-composition on the AES core

Showcase of flow for RISC-V processor CV32E40P

(case study mixing both hardware and software)

1. Scripts structure
2. Flow launch



Conclusion: µArchiFI a pre-silicium formal tool for FIA robustness

Key take-aways

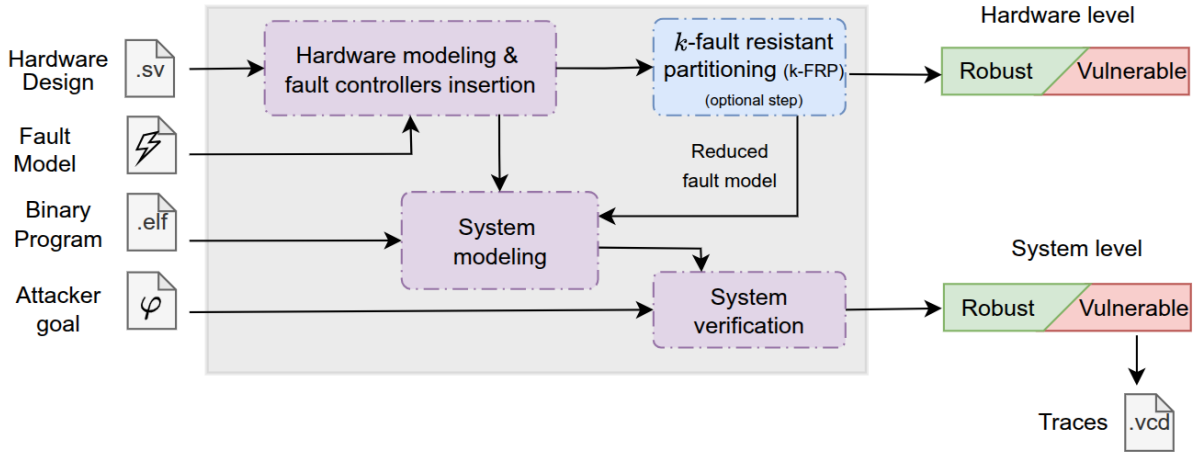
- Better understanding of propagation of faulty behaviors & prove robustness of HW/SW
- Reduce design costs/delays (e.g. prove SW countermeasures to avoid HW respin) & increase confidence before certifications
- Exhaustive methodology, configurable fault model and RTL/netlist levels analyses, agnostic to EDA flows

New release v0.2, feedback/contributions are welcomed!

- New functionalities: new fault model, iterator of cex, container, tests and bug fixes
- Tested designs: CV32P and Ibex

Work-in-progress

- Central configuration file
- Testing examples based on CV32S

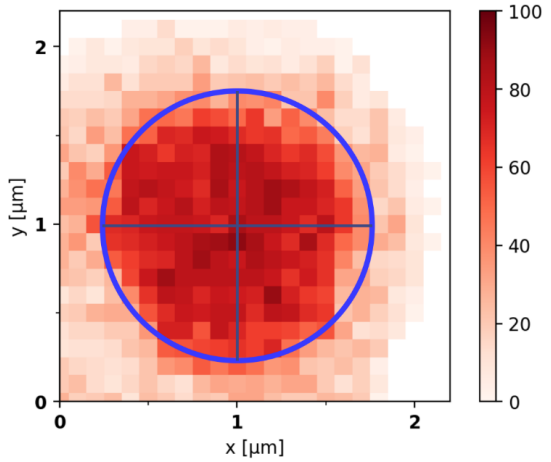


(Some) R&D next steps for µArchiFI

Compositional verification of multiple countermeasures

Leveraging layout information to design new fault models (not within FORWARD)

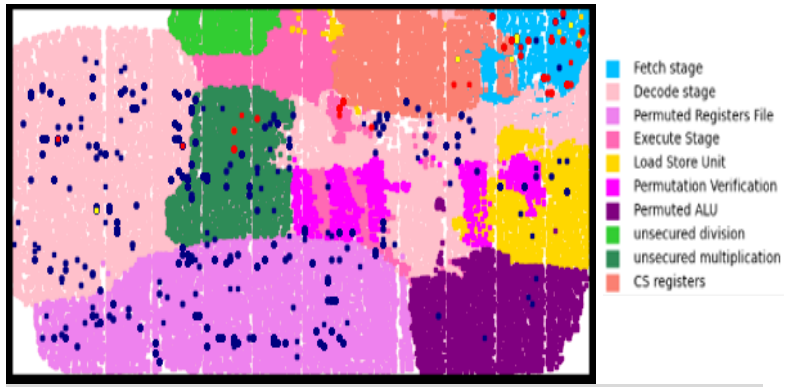
- A laser spot may fault several neighbour cells:
  → Select signals according to location constraints of laser spot
- Extracted from physical models
  - Exclude multi-faults configuration / fault models applied over all bits within this area



Laser characterization, 28nm bulk CEA Leti, L. Mangin (2025)

Compare results between experimental fault injection campaigns with formal verification

- Compare laser injections results over VASCO with formal analyses (non secure and secure AES + secure proc.)
- Support clock glitch fault model in pre-silicon verification



Mapping of laser-induced faults affecting a circuit (VASCO#2 – FD22nm, CV32E40P processor).

Thank you

Questions?

Web site of µArchiFI: **https://list.cea.fr/fr/page/µarchifi/**

Github of µArchiFI: **https://github.com/CEA-LIST/uArchiFI**

Publications:

[FDTC 2022] S. Tollec et al. “Exploration of Fault Effects on Formal RISC-V Microarchitecture Models,” in FDTC, 2022.
[FMCAD 2023] S. Tollec et al. “ARCHIFI: Formal Modeling and Verification Strategies for Microarchitectural Fault Injections,” in FMCAD, 2023.
[CHES 2024] S. Tollec, V. Hadžic, P. Nasahl, M. Asavoae, R. Bloem, D. Couroussé, K. Heydemann, M. Jan, and S. Mangard “Fault-Resistant Partitioning of Secure CPUs for System Co-Verification against Faults,” IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES), 2024.
[HOST 2026] J. Alle Monne et al. “Synthesis of RTL-based Characterization Programs for Fault Injection”, in HOST, 2026, To appear.

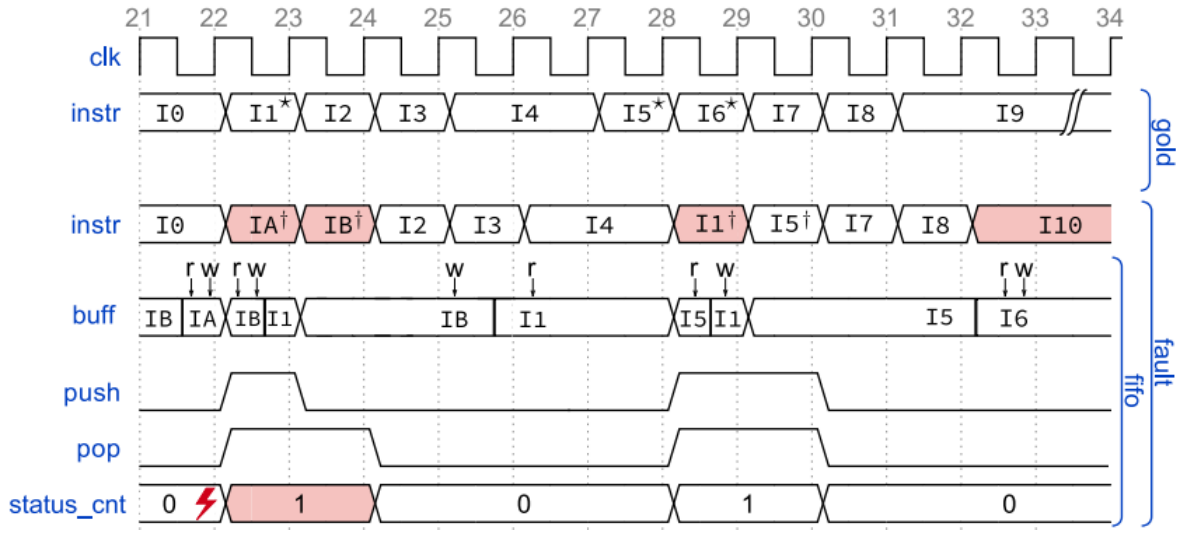
Pre-silicon competitors vs. µArchiFI

	System	Analysis level	Evaluation technique	Properties
EDA formal tools SIEMENS cadence	HW	All (RTL, netlist, P&R)	Formal	Functional
ISO26262-oriented tools SYNOPSYS OPTIMA	HW	All (RTL, netlist, P&R)	Simulation	Reliability
SCA-oriented tools SECURE-IC KEYSIGHT	HW/SW	All (RTL, netlist, P&R)	Simulation	Extra-functional (Timing, power, electromagnetic emission)
FIA-oriented tools CASPIA FortifyIQ OPTIMA	HW/SW	RTL	Simulation	Extra-functional
µArchiFI	HW/SW	**RTL & netlist (P&R targeted)**	**Formal**	Extra-functional

Results using µArchiFI

Vulnerability identification & fault effects: cannot be modeled at ISA level & provide valuable information to processor designer

Formal proof of countermeasures

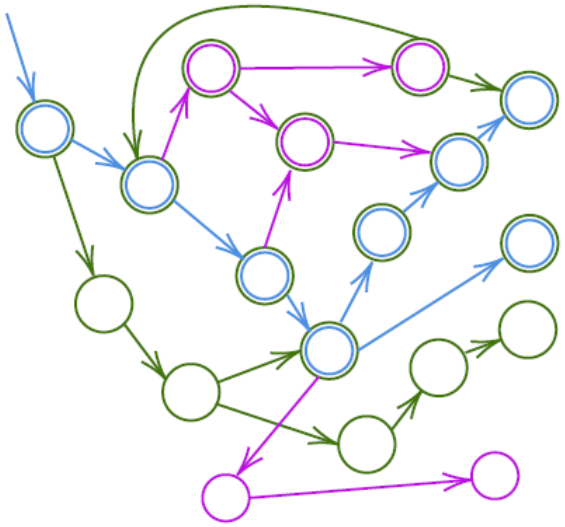


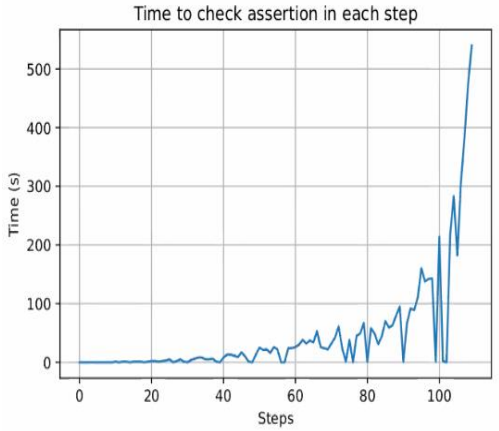
Performance summary

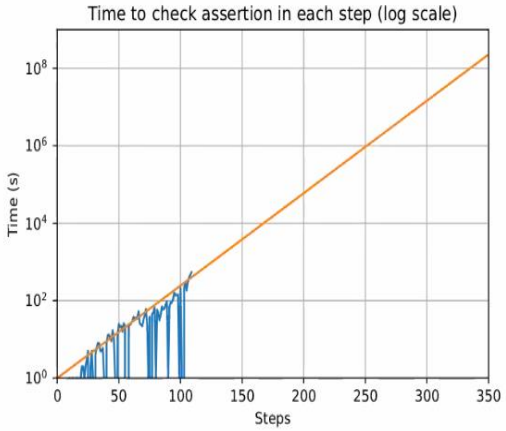
- Small in-order CPU
- 1 fault injection (from a single bit-effect to symbolic effects)
- Verification using **Bounded Model Checking** (BMC)
- → limited unrolling (~ 100 instructions)

Use case names		*I - Robust Software*	*II - Robust Hardware*	*III - Cryptographic Software*
Hardware design	name:	**CV32E40P (Riscy)** - RISC-V - 4 stages	**Secure Ibex** - RISC-V - 2 stages - dual core	**Ibex** - RISC-V - 2 stages
	gates:	2842	4422	1983
	FFs:	179	211	114
	size*(GE):	89954	61452	26327
Software program		VerifyPIN_V7 [Dur+16]	VerifyPIN_V1 [Dur+16]	Key Schedule (AES) [kok19]
Attacker Goal φ		Bypass authentication without triggering SW alert	Bypass authentication without triggering HW alert	Set to 0 a byte in the penultimate round key
Fault model $\mathcal{F}$	location:	Sequential logic Control Path	Sequential logic Redundant CPU Core	Combinational logic Execute stage of CPU
	effect:	Symbolic	Symbolic	Reset
	timing:	60:*	*	*
Number of FIs N		1	5	2
BMC depth k		75	46	38
Verification results		φ is reachable	φ is unreachable	φ is unreachable

Challenge: state explosion bottleneck







Step 10: 10 sec.
Step 50: 4 min. 20s
Step 100: 1h 12 min.

Step 150: 18h50
Step 200: 112 days 6h
Step 350: 130 years

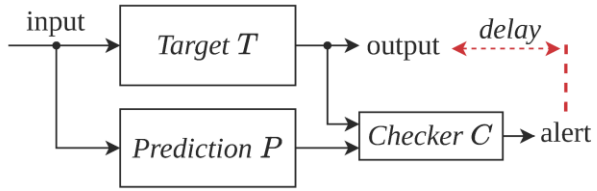
Specialized approaches are needed! (k-FRP)

- **Large HW designs, large programs (SW)**
  - Focus on specific parts to be analysed, simulation on other parts…
- **Faults incur extra analysis complexity**
  - In particular, multiple faults → restrict areas to be faulted
- **Performance (without k-FRP step)**
  - Small in-order CPU (46 kGE)
  - ~ 100 instructions
  - 1 fault injection (from a single bit-effect to symbolic effects)

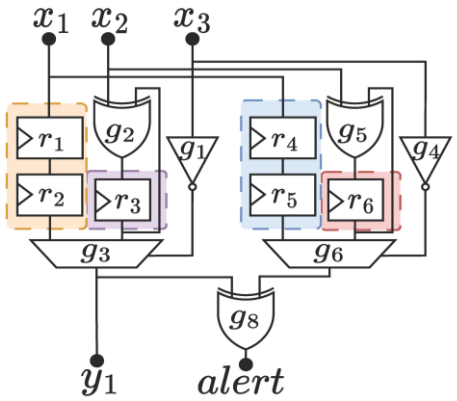
k-Fault Resistant Partitioning [CHES, 2024]

Intuition

Concurrent Error Detection Scheme



Partitioning example with *k = 1*



Theorem

k-fault resistant partitioning ⇒ k-fault security

Validation

Research Objectives

- No CPUs analysis tools or benchmarks available for comparison
- Evaluate verification performance
- Consider multiple-fault attacks
- Compare with prior work like FIVER [RR21]

Impeccable Circuits [AM19]

- Designed to detect up to 3 faults (up to 7 faults for AES)

Test case	# faults	FIVER	kFRP (ours)
AES	2	130 h	4 h
AES	3	∞	55 h **(*)**
(*) we identify exploitable faults in the checker of Skinny and AES			

[AM19] Aghaie, Anita, et al. "Impeccable circuits." *IEEE Transactions on Computers* (2019)
[RR21] Richter-Brockmann, Jan, et al. "Fiver–robust verification of countermeasures against fault injections", *CHES 2021*

Case study: analysis of a Secure Element

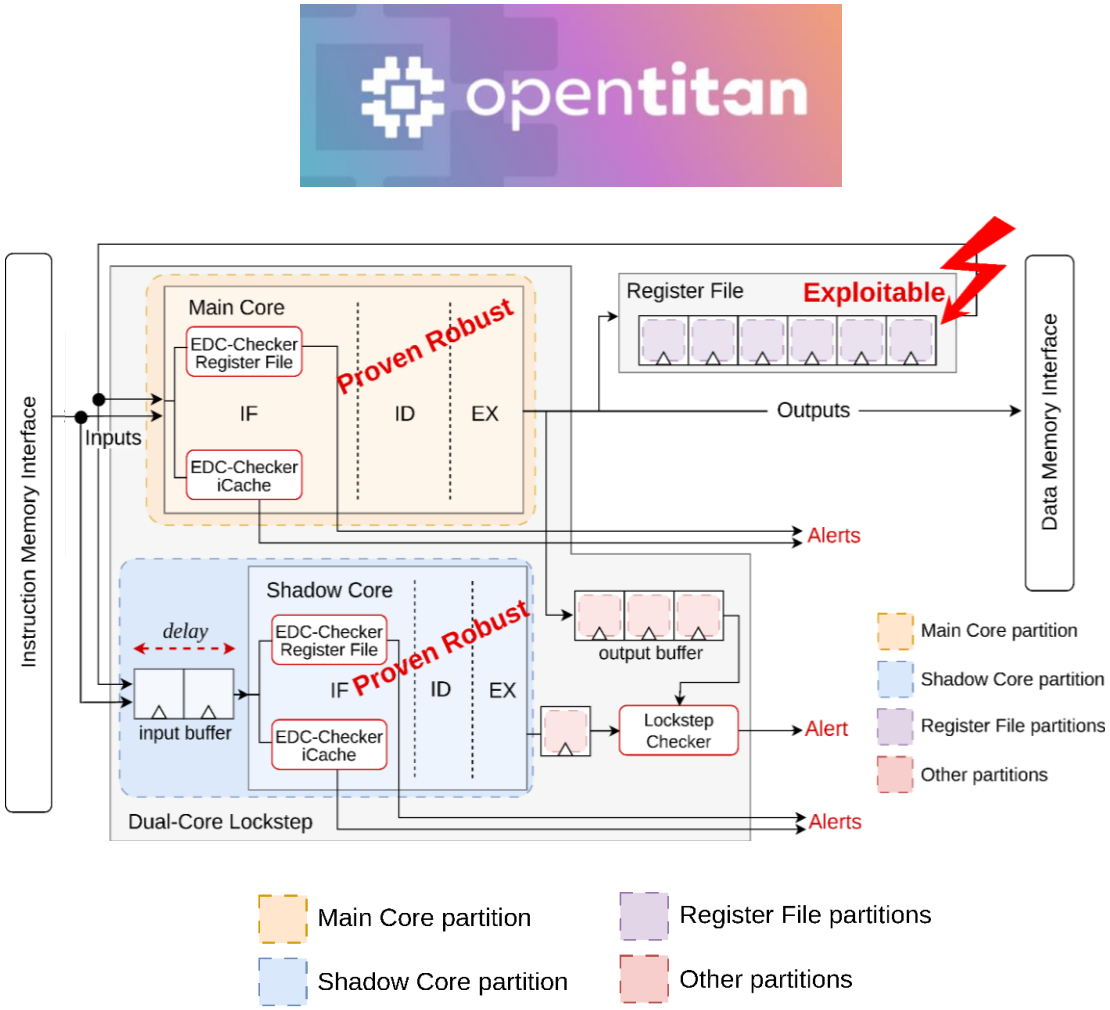
OpenTitan: open-source Root of Trust [JR18]

- TRL 8 according to LowRISC
- Secure-Ibex RISC-V processor [Lo18]
  - Embeds several HW countermeasures: Dual Core Lockstep (DCLS), Error Detection Codes in Register File

Results of applying our pre-silicon methodologies

- Fault model: single transient bit-flip everywhere at any time
- **Vulnerability Reported:** 172 exploitable faults — allow reading from an incorrect register location
  - We proposed a security fix and formally prove it using our methodology (**integrated into the OpenTitan project)**

- **Secure Ibex** is now **proven 1-fault secure** unconditionally of the executed software
  - Prove 1-fault security (DCLS + Error detection codes) in 68h



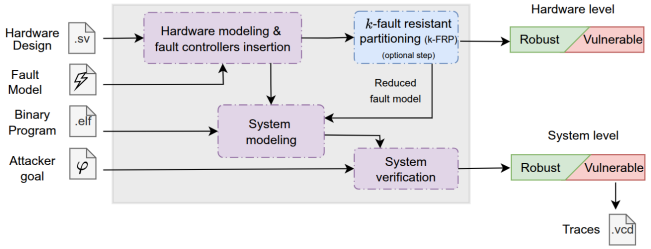
[JR18] Johnson, Scott, et al. "Titan: enabling a transparent silicon root of trust for cloud." *Hot Chips: A Symposium on High Performance Chips*. Vol. 194. 2018.
[Lo18] lowRISC. Ibex RISC-V Core github repository. https://github.com/lowRISC/ibex. Accessed: February 22, 2024.

k-FRP's scalability improvements

Evaluation of the Secure Ibex and its modules using k-fault-resistant partitioning

Circuit Characteristics			Faults		Algorithm 1 Performance			Results		
Name	Size (GE)	Regs (#)	Loc. (#)	Order k	*BuildPartitioning*		*CheckIntegrity* Time	Partitions (#)	Exploitable Faults	
					Iter. (#)	Time			$\mathcal{P}'$ (#)	$\mathcal{F}'$ (#)
Register File	12 075	1 326	8 331	1	172	38 s	53 s	1 326	0	172
			1 326[a]	3	1	349 s	344 s	1 326	0	0
Register File with fix	11 913	1 326	8 667	1	1	17 s	73 s	1 326	0	0
			1 326[a]	3	1	135 s	383 s	1 326	0	0
DCLS	117 998	5 918	116 561	1	508	20 h 12	5 h 10	1 108	0	0
				2	11	11 s	—	445	—	—
Secure Ibex (no iCache)	130 194	7 248	125 080	1	1	10 h 45	30 h 50	2 438	0	0 (+172)
				2	48	53 s	—	421	—	—

[a] Restricted fault model targeting the sequential logic only



µArchiFI	kGE (#)	faults (#)	CPU cycles (#)	analysis level
Without k-FRP	20-90	1 (* ≤5)	≤100	RTL
With kFRP	130	≤3	unlimited	netlist