



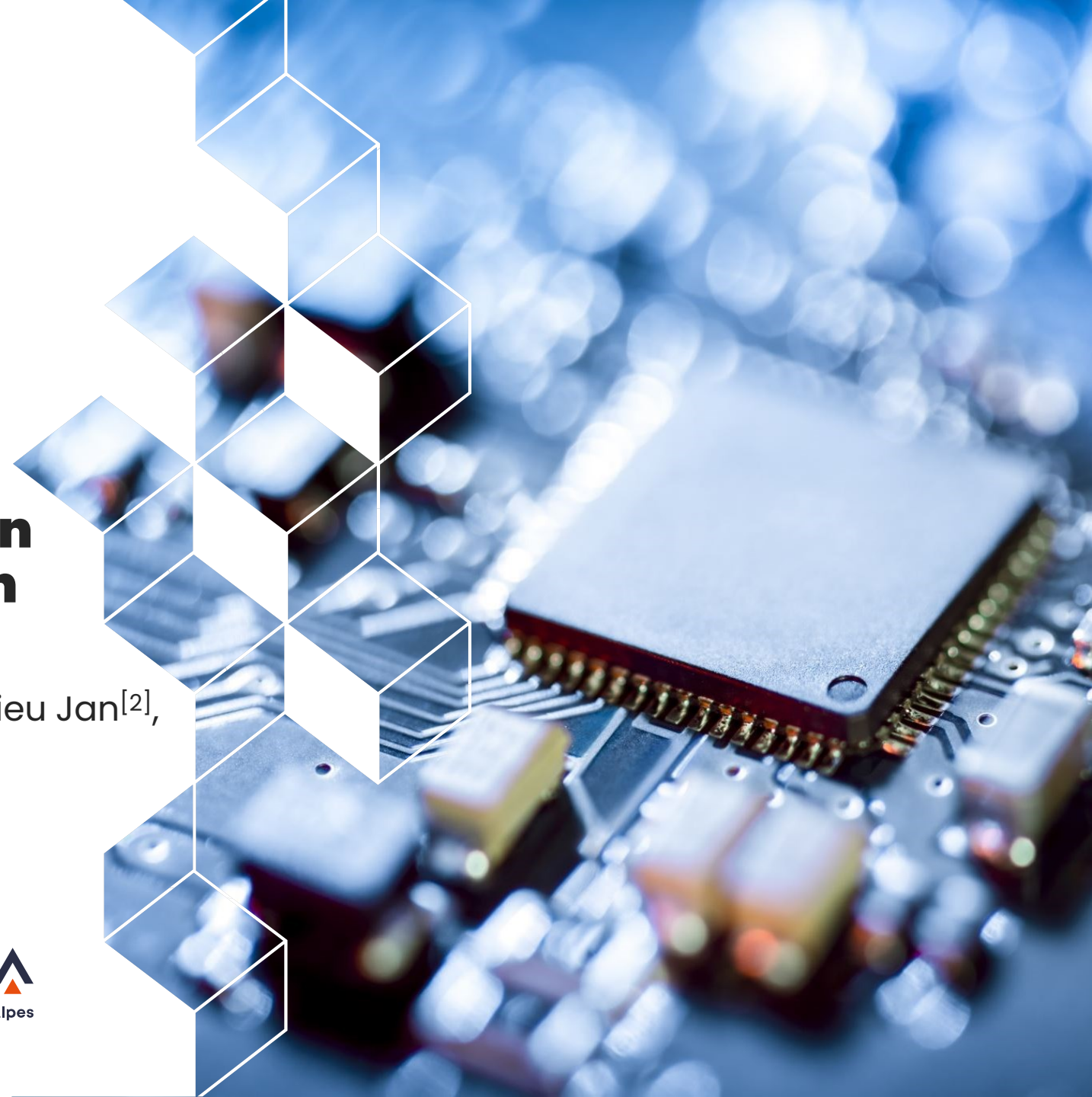
Bridging Laser Fault Injection Experiments and Pre-Silicon Analysis on AES Hardware

Giorgio Fardo^[1], Damien Couroussé^[1], Mathieu Jan^[2],
Laurent Maingault^[3], Loïc Mangin^[3]

[1]: Univ. Grenoble Alpes, CEA, List, F-38000, Grenoble, France

[2]: Univ. Paris-Saclay, CEA, List, F-91120, Palaiseau, France

[3]: Univ. Grenoble Alpes, CEA, Leti, F-38000, Grenoble, France





1. Background

Fault Injection: the need for pre-silicon analysis

- ❑ Traditional post-silicon evaluation is costly and time consuming, needs tape-out, dedicated instrumentation
- ❑ Fault Injection attacks: voltage/clock/EM/laser; LFI is standard practice in security evaluation labs
- ❑ Three structural limits: no exhaustiveness over the fault space, limited internal observability, feedback arrives after tape-out





2. Goal

Research question

- ❑ **Can pre-silicon analysis recover signal candidates leading to observed faulted ciphertexts?**

Case study:

- ❑ Bridge a real LFI campaign on an ASIC AES-128 accelerator with formal pre-silicon analysis

Contributions:

- I. Partitioning strategy + parallel execution engine
- II. Cross-validation methodology against silicon results
- III. Fault-model insights on multi-byte ciphertexts

Pre-silicon analysis: simulation vs formal

Simulation based RTL/gate

- ❑ Cheap
- ❑ Traces cover only concrete inputs
- ❑ Difficult to find fault locations that generate a specific output (**reverse analysis**)

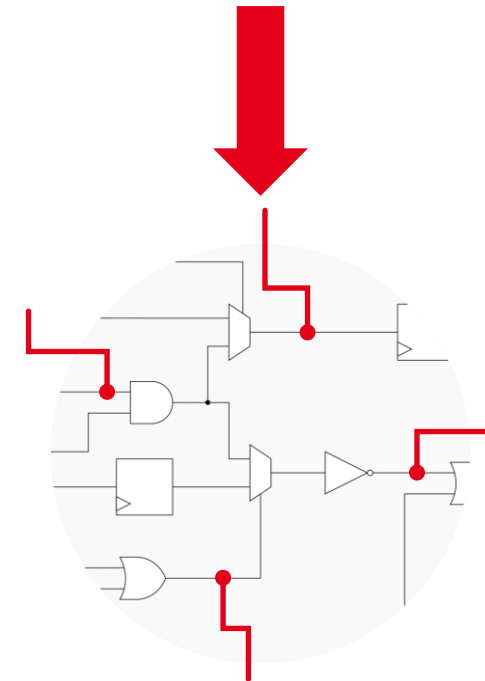
Formal

- ❑ **Exhaustive** under the stated fault model
- ❑ Can **prove** robustness or exhibit corner cases
- ❑ Easier **reverse analysis** due to support of symbolic variables

Faulted
Chipertext

96f580a57d4630c9e52deb01

Fault signal
candidates

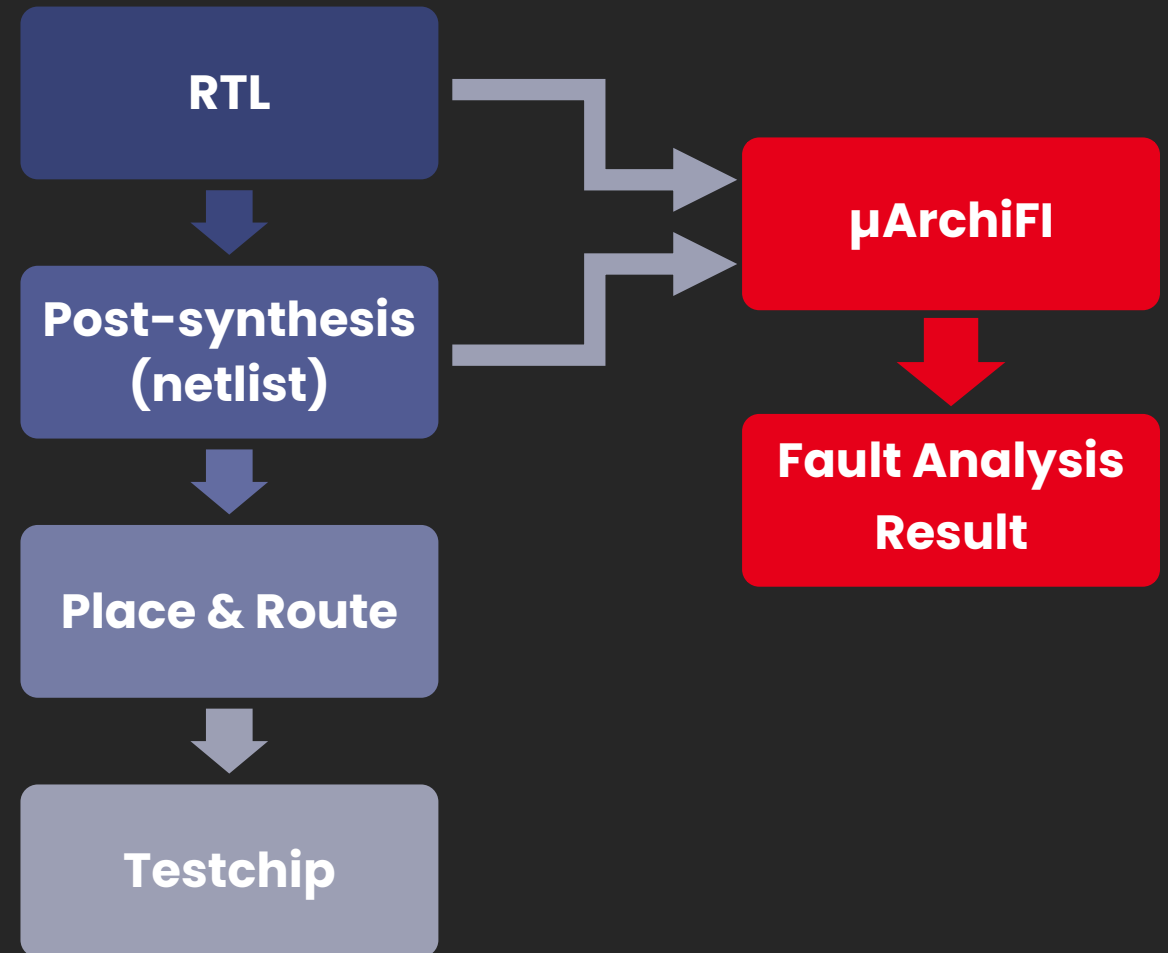




3. **μArchitectFI**

μArchiFI^[1]

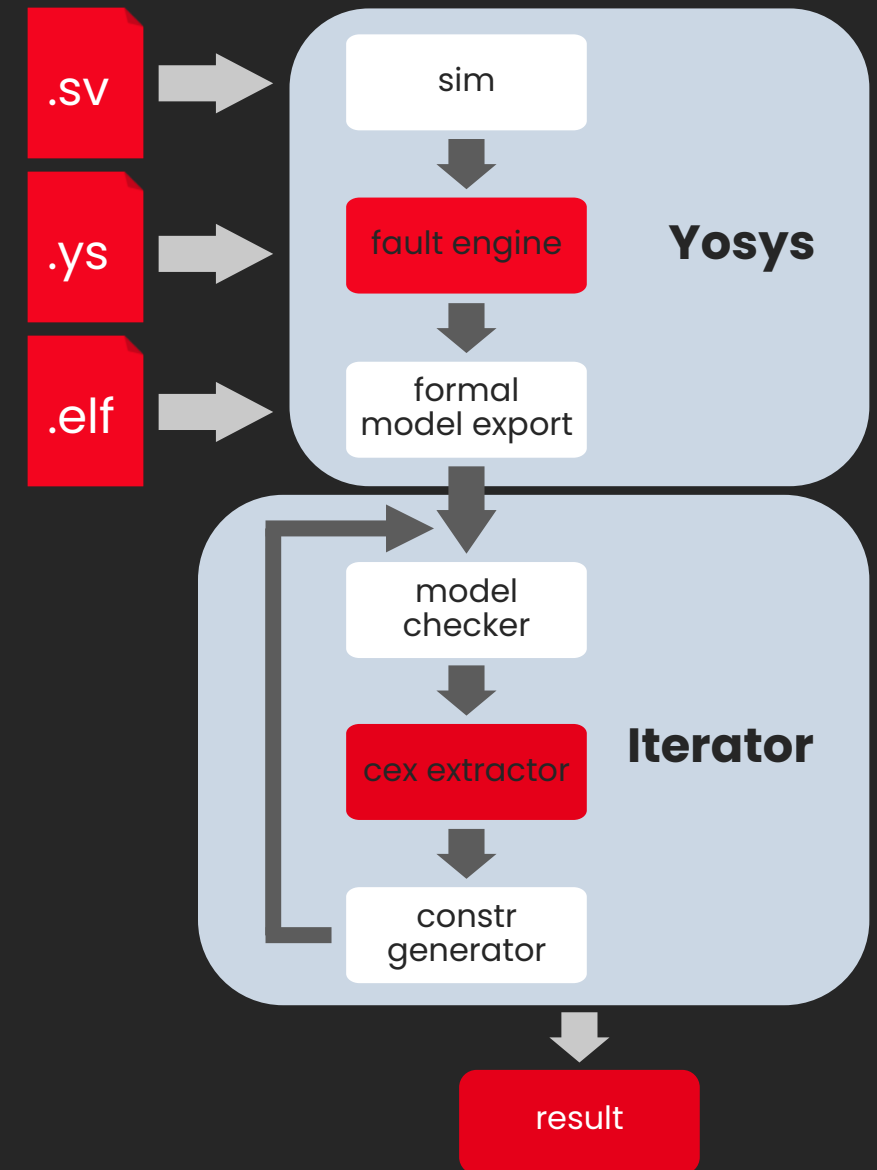
- ❑ Fault injection analysis tool
 - ❑ Open source
 - ❑ Formal model checking based
 - ❑ Can do Hardware or Hardware + Software co-verification
 - ❑ Configurable fault model: location, timing, fault order
- ❑ Prove robustness of (HW/SW) countermeasures to secure a system
- ❑ Identify fault effects at the HW level and check whether it generates vulnerabilities at SW level



[1] <https://github.com/CEA-LIST/uArchiFI>

μArchiFI internal architecture

- ❑ Yosys processing:
 - ❑ Design simulation to bootstrap analysis
 - ❑ Insertion of fault-injection muxes on selected signals
- ❑ Formal model generation:
 - ❑ Leveraging off the shelf model checkers: Yosys-smtbmc, Pono
 - ❑ Iteration based on auto constructed constrains from VCD CEX
 - ❑ Bounds based on time and # of iterations

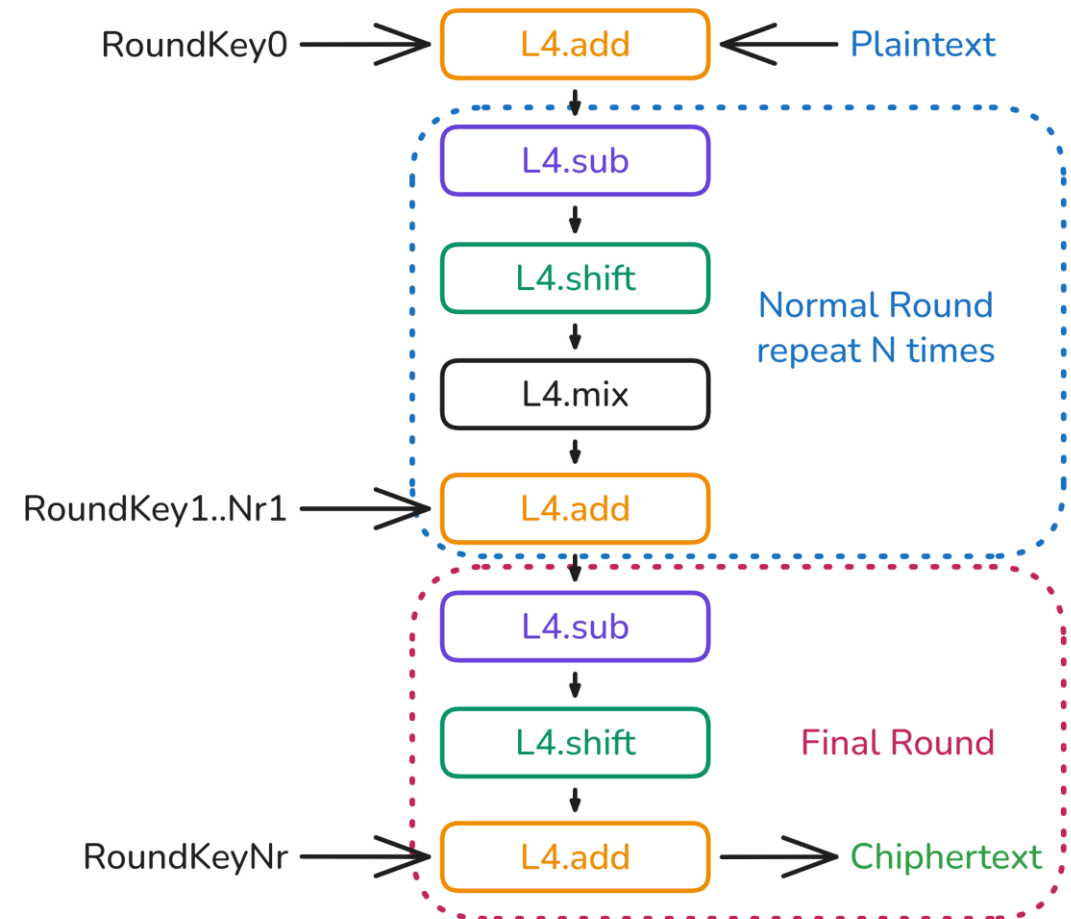
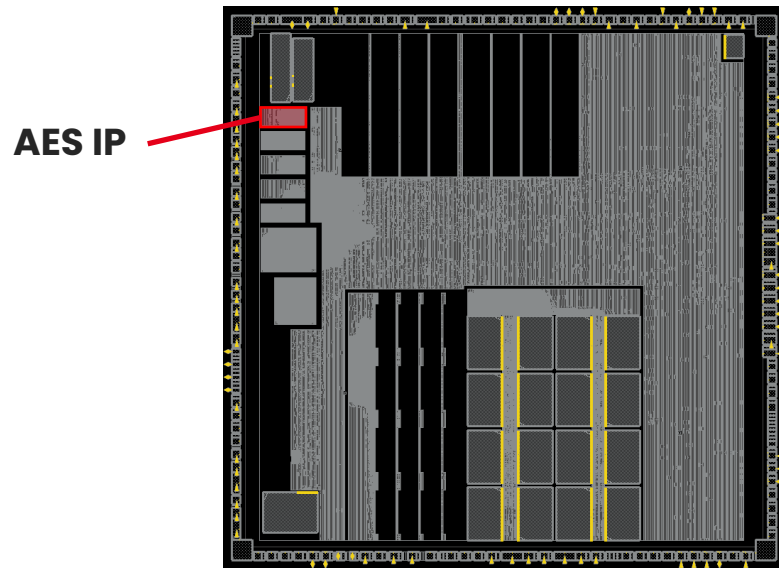




4. The test case

Target: VASCO-2^[1] AES accelerator

- ❑ FD-SOI 22nm
- ❑ Internal evaluation test chip
- ❑ Target the AES 128 bit in ECB mode
- ❑ No protections/hardenings in place



[1] M. Carmona, S. Di Matteo, F. Lepin, M. Pezzin, E. Valea, and R. Wacquez, "VASCO: ASIC test platform for hardware security research," Mar. 2025. [Online]. Available: <https://cea.hal.science/cea-05026004>

LFI Campaign^[1]

- ❑ Backside injection, ~1.5 µm spot, 399 fault injection parameters x 360 shots = 143,640 shots
- ❑ ~ 90% no effect
- ❑ Nine exploitable faulty ciphertexts:
 - ❑ Rounds 4–7

Round	Ciphertext	Reps
5	96f580a57d4630c9e52deb013cb363f3	5858
5	2162fd6cdc05ea1d472a2f70e9c7e246	3634
6	e1a69eb8d9504331e7e9d0b68aaebdb5	2560
6	b1138e1fe7ff69a8798d9efd0b4d6f4b	959
4	f7b80e880e55143ca6c6f8ec64139150	869
7	9f0225151b4e7bc6af51d1d1e29963a5	100
7	825dc7f13503d70db16ee9cb1f670e0a	34
7	6e39fc521a5429c4842af50c5460f7c0	12
4	ca7d05612a6018fa665e178b4d697575	10

Ciphertext targets

[1] L. Mangin *et al.*, "FD-SOI rather than Bulk - Experimental investigation of laser induced fault mechanisms in FD-SOI," *2026 IEEE 32nd International Symposium on On-Line Testing and Robust System Design (IOLTS)*, Polignano a Mare, Italy, 2026, pp. 1–7, doi: 10.1109/IOLTS69666.2026.11633739.

Algorithmic analysis

- ❑ Start from target ciphertext
- ❑ Decrypt with base key
- ❑ Compare with normal encryption trace to find minimized hamming (**HD**) distance based injection point
- ❑ Extract number of bit/bytes faulted
- ❑ Discriminate into two groups

Round	Num bit	Ciphertext	HD
4	1	f7b80e880e55143ca6c6f8ec64139150	1
5	1	96f580a57d4630c9e52deb013cb363f3	1
6	1	e1a69eb8d9504331e7e9d0b68aaebdb5	1
7	1	825dc7f13503d70db16ee9cb1f670e0a	1
7	1	6e39fc521a5429c4842af50c5460f7c0	1

Single bit targets

Round	Num Bytes	Ciphertext	HD
4	5	ca7d05612a6018fa665e178b4d697575	16
5	5	2162fd6cdc05ea1d472a2f70e9c7e246	17
6	5	b1138e1fe7ff69a8798d9efd0b4d6f4b	18
7	5	9f0225151b4e7bc6af51d1d1e29963a5	17

5 Byte targets

Formal setup: how a ciphertext becomes a property

- ❑ Same plaintext and key as FIPS test-bench
- ❑ Assertion: `data_out_tb /= ciphertext`
- ❑ A counterexample (CEX) is generated if at the end of the encryption the output is the **same** as the **expected faulted ciphertext**
- ❑ One fault injected per experimental round
- ❑ CEX: fault reproducing that exact target ciphertext

```
1 process(clock_bench, data_out_tb, done_tb)
2 begin
3   if rising_edge(clock_bench) then
4     if done_tb = '1' then
5       assert data_out_tb /= ciphertext;
6     end if;
7   end if;
8 end process;
```

Analysis parameters

❑ **Fault models:**

- ❑ Single-bit flip: flips a single bit in the target signal (**xor**)
- ❑ Symbolic: solver tries all possible values of the target faulted signal (**any**)

❑ **Fault parameters:**

- ❑ 1 fault per run, single cycle activation

❑ **Simulation:**

- ❑ Simulate up until 1 cycle before target round

❑ **Formal:**

- ❑ BMC depth to end of AES encryption

❑ **Analysis time:**

- ❑ Timeout after 120 hours of wall time

❑ **Coverage:**

- ❑ Defined as the percentage over the total signals faulted that generated a CEX or passed the solving process with no CEX generated

Research questions

- ❑ **RQ1:** Tractability
 - ❑ Is the reverse analysis doable with the default μ ArchiFI flow?
- ❑ **RQ2:** Single-bit
 - ❑ Can we reproduce the single-bit faulted ciphertexts?
- ❑ **RQ3:** Multi-byte fault model
 - ❑ Can we outright exclude some fault models as candidates?
- ❑ **RQ4:** Dependency on target ciphertext
 - ❑ Can we exclude ciphertext dependency of the signal candidates using an unconstrained analysis?



5. Untractability

The untractability problem

- ❑ Monolithic selection: all signals, both fault models, all 9 ciphertext candidates
- ❑ Takeaway framing: **partitioning is a prerequisite**, not an optimization

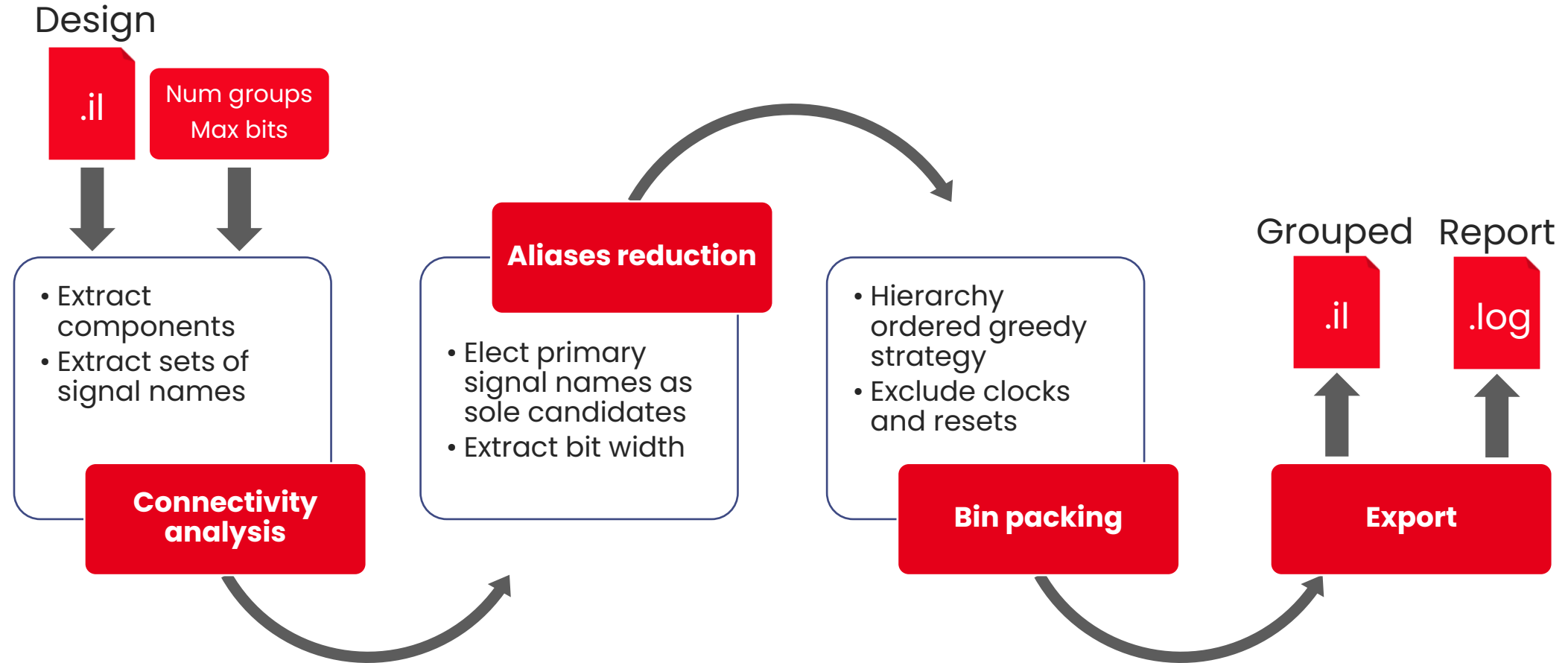
Target	Fault model	Wall (h)	CPU (h)	Mem (GB)	1° CEX (s)	# CEX	Cov (%)
1 bit	1 bit	120	600	16	–	0	0
1 bit	Symbolic	120	600	17	–	0	0
5Byte	1 bit	120	600	15	–	0	0
5Byte	Symbolic	120	600	16	–	0	0

No partitioning results

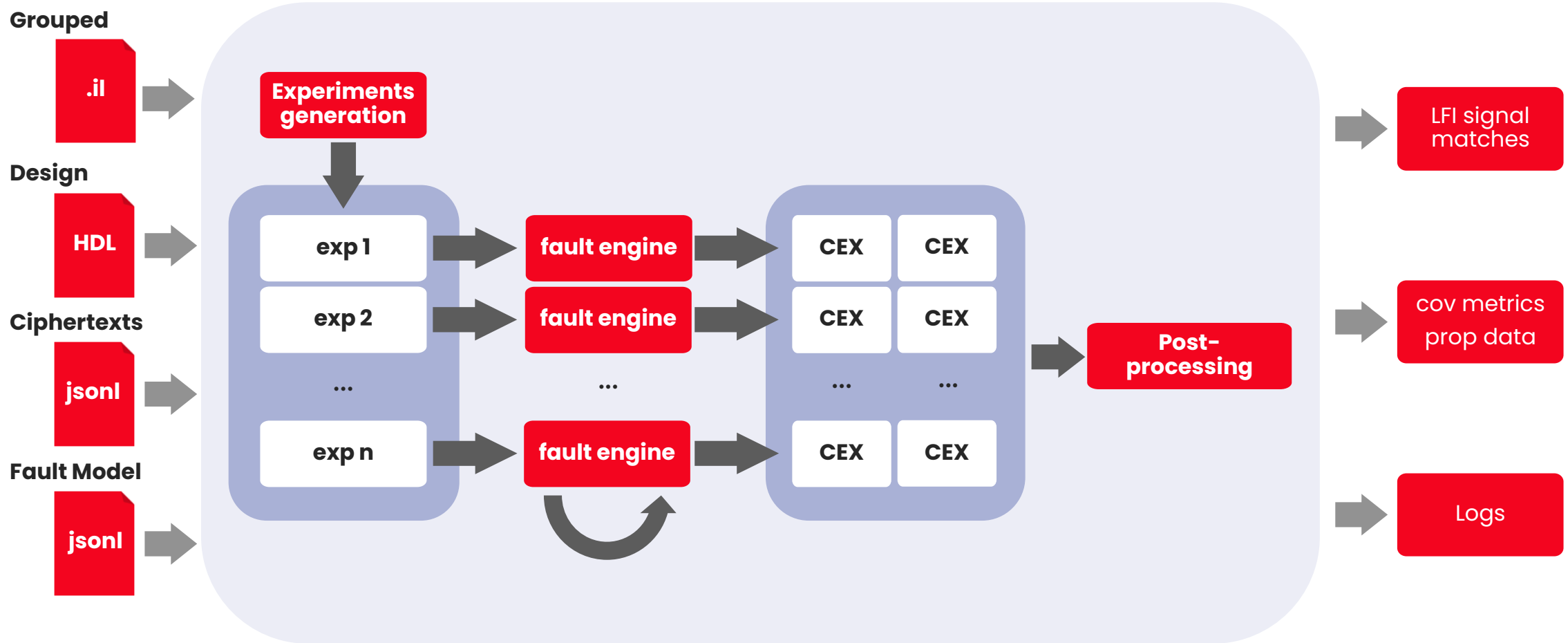


6 ■ **μArchFI extensions**

Extension 1: Design-aware partitioning



Extension 2: Multi-manager execution engine





7. Results

Targets recap



Single bit targets

Round	Num bit	Ciphertext	Count
4	1	f7b80e880e55143ca6c6f8ec64139150	869
5	1	96f580a57d4630c9e52deb013cb363f3	5858
6	1	e1a69eb8d9504331e7e9d0b68aaebdb5	2560
7	1	825dc7f13503d70db16ee9cb1f670e0a	34
7	1	6e39fc521a5429c4842af50c5460f7c0	12

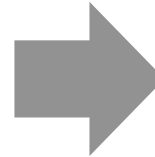
Five Byte targets

Round	Num Bytes	Ciphertext	Count
4	5	ca7d05612a6018fa665e178b4d697575	10
5	5	2162fd6cdc05ea1d472a2f70e9c7e246	3634
6	5	b1138e1fe7ff69a8798d9efd0b4d6f4b	959
7	5	9f0225151b4e7bc6af51d1d1e29963a5	100

RQ1: What partitioning buys

No partitioning:

CEX	Wall time	Coverage
0	120 h	0 %
INTRACTABLE		



Design-aware partitioning:

CEX	Wall time	Coverage
28	27 h	~ 95 %

RQ1: What partitioning buys



Class	Fault Model	Wall (h)	CPU (h)	Mem (GB)	1° CEX (s)	# CEX	COV ranges %
1 bit	xor	27	126	6	2	28	93-100
1 bit	Symbolic	27	6	16	1	43	89-95
5 Bytes	xor	25	106	5	–	0	89-100
5 Bytes	Symbolic	25	21	12	1	28	89

Design-aware partitioning results

RQ2: Single-bit targets

- ❑ **Fault model:** single-bit flip **28** CEX; symbolic **43** CEX
- ❑ Vulnerable classes: *AddRoundKey* (add, sig_add_out), *MixColumns* (mix, temp_mix), round state mux (sig_mux_out), byte substitution, round-key interfaces

Signal	Ciphertext Round
add.no1	6
mix.mix2	4, 6
mix.temp_mix	4-7
sig_add_out	4-7
sig_mux_out	4-7
sig_temp	4-7
sig_roundkey	4-7

Single-bit flip fault model signal candidates

Signal	Ciphertext Round
add.no1	5,7
byte.sig_byte	4-7
mix.mix2	5
mix.temp_mix	4-7
sig_add_out	4-7
sig_byte_out	4-7
sig_mux_out	4-7
sig_shift_out	4-7
sig_temp	4-7
sig_roundkey	4-7

Symbolic fault model signal candidates

- ❑ **Signal candidates for the 1 bit targets can be found**

RQ3: Five-byte targets

- ❑ **Fault model:** single-bit flip, one cycle: **zero** CEX across all four targets, at 89–100 % coverage
- ❑ Symbolic, one-cycle: **28** CEX

Symbolic		
Round	Signals verified	Cov (%)
7	295	89
6	295	89
5	295	89
4	295	89

Symbolic fault model coverage

Signal	Ciphertext Round
byte.sig_byte	4-7
sig_add_out	4-7
sig_byte_out	4-7
sig_mux_out	4-7
sig_shift_out	4-7
sig_temp	4-7
sig_roundkey	4-7

Symbolic fault model signal candidates

- ❑ **Single-bit fault model cannot reproduce the 5 Bytes targets**



8 **Conclusions & future** **work**

Conclusions

01

TRACTABILITY

Partitioning **enables** the reverse analysis.

02

SIGNAL CANDIDATES

LFI fault model is **reproducible** in pre-silicon analysis

03

FAULT MODEL

Fault model discrimination on **multi-byte** effects.

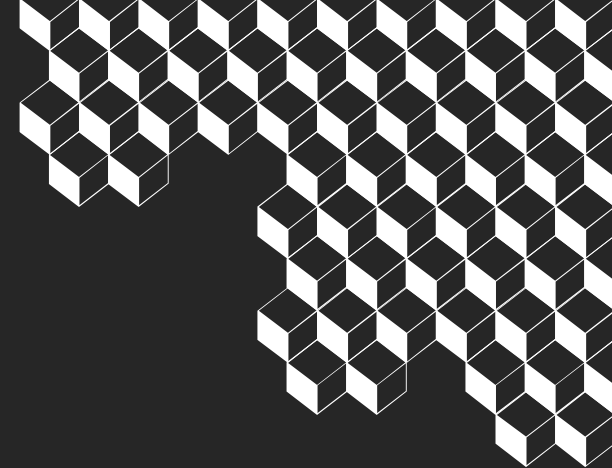
Future work

- ❑ RTL to layout mapping vs laser coordinates
- ❑ Intermediate and multi-cycle fault models
- ❑ Protected AES designs

μArchiFI is open to contributions & collaborations!!



<https://github.com/CEA-LIST/uArchiFI>



Thanks

Giorgio Fardo
giorgio.fardo@cea.fr





9. **Extra slides**

RQ4: Unconstrained analysis

- Unconstrained analysis (assert output $\neq$ golden): 898 / 926 CEX in under an hour
- Every signal from the reversing campaigns also appears there

Target	Wall (h)	CPU (h)	Mem (GB)	1° CEX (s)	# CEX	COV ranges %
unc_fault_xor	<1	<1	1	1	898	100
unc_fault_any	<1	<1	1	1	926	100