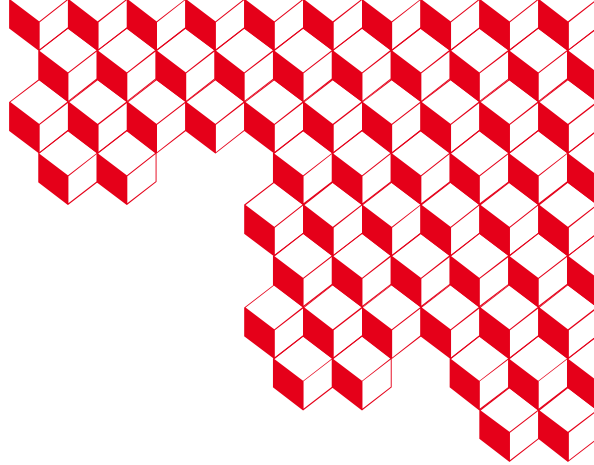


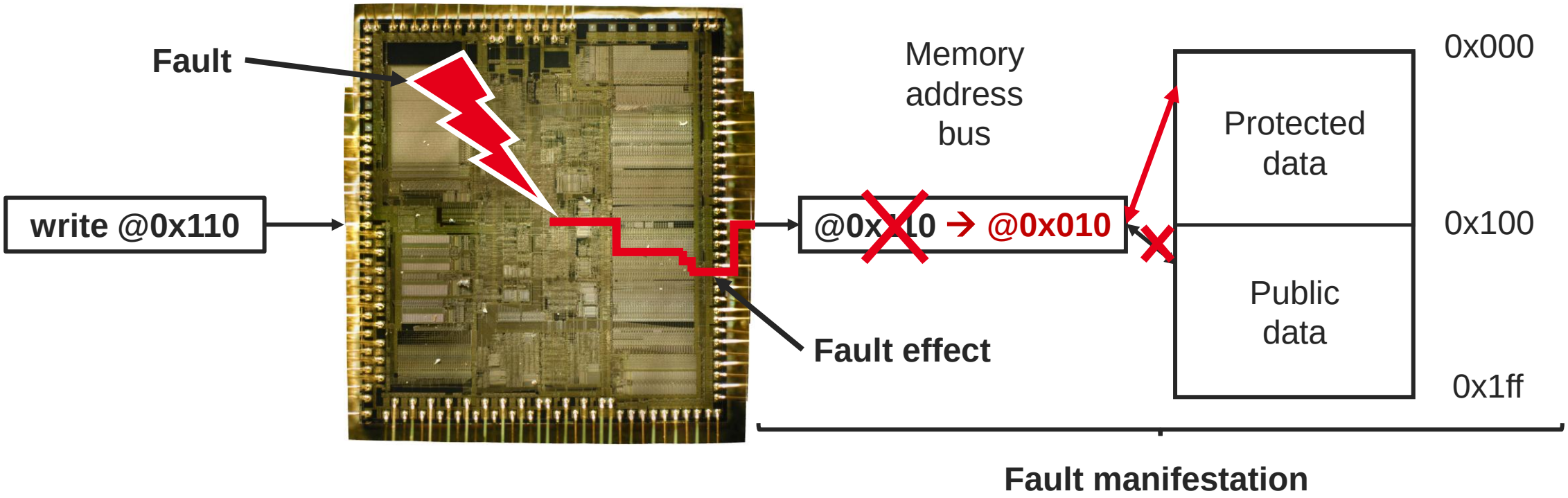
Synthesis of RTL-based Characterization Programs for Fault Injection

Jonah Alle Monne (CEA), Guillaume Bouffard (ANSSI), Damien Couroussé (CEA), Mathieu Jan (CEA)

HOST26 Presentation

Fault injection attacks on CPU

- Fault injection = Intentionnaly deviate a device from intended behavior using physical perturbations (Laser, Electromagnetic, Voltage…)
- Cryptographic circuits, smartcards, bootloaders… are vulnerable.

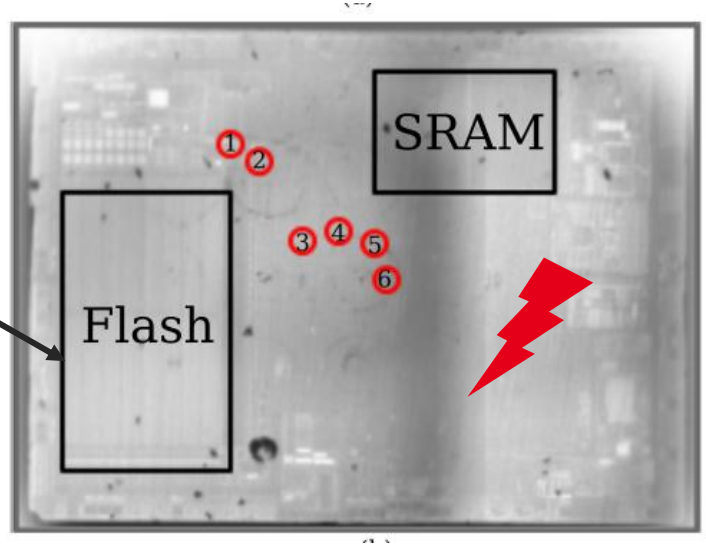


- No observability → need for analysis of faults effects
- **Problem**: Understanding the **fault** and the **fault effect** with **fault manifestation** is hard

Characterization tests

Objective: understand where the fault has been injected

```
i1:   add r0, r0, 0x1
i2:   add r0, r0, 0x2
i3:   add r0, r0, 0x3
i4:   add r0, r0, 0x4
i5:   add r1, r1, 0x1
i6:   add r2, r2, 0x1
i7:   add r3, r3, 0x1
i8:   add r4, r4, 0x1
i9:   add r0, r0, 0x5
i10:  add r0, r0, 0x6
i11:  add r0, r0, 0x7
i12:  add r0, r0, 0x8
```



Extracted from [khuat, 2021]

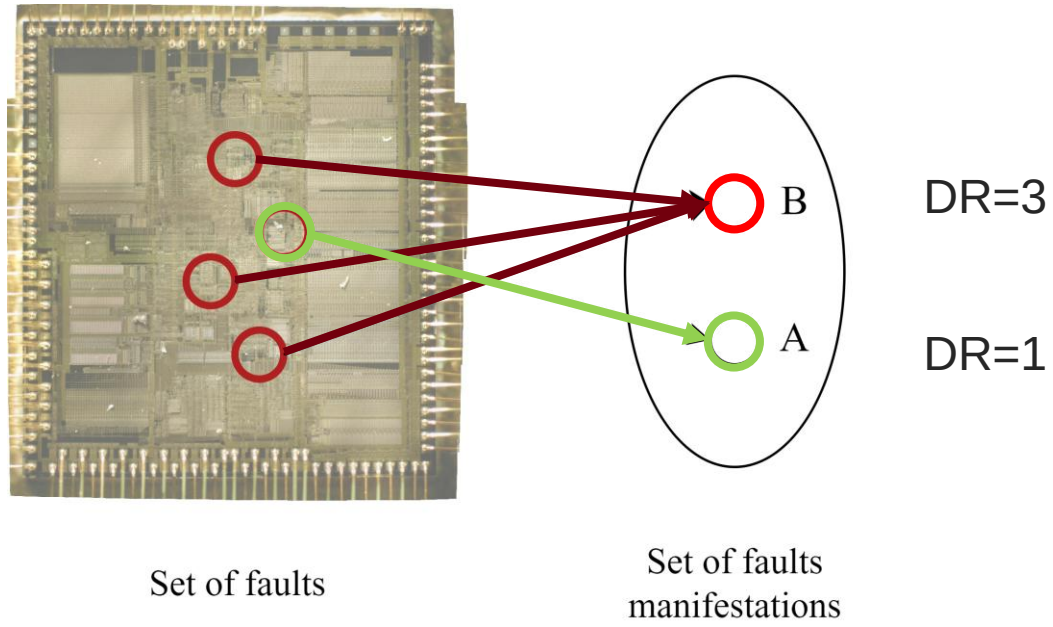
1. **Run a characterization program** on the target
2. Randomly **inject faults**
3. **Observe** the register file output
   CRASH
   NO FAULT MANIFESTATION
   FAULT MANIFESTATION
4. Try to link **fault and fault effect** to each **fault manifestation** with educated guess

Problem: several **faults** can lead to the same **fault manifestation**

[khuat, 2021] - Vanthanh Khuat, Jean-Luc Danger, Jean-Max Dutertre. Laser Fault Injection in a 32-bit Microcontroller: from the Flash Interface to the Execution Pipeline. 2021 Workshop on Fault Detection and Tolerance in Cryptography (FDTC)

The discrimination problem

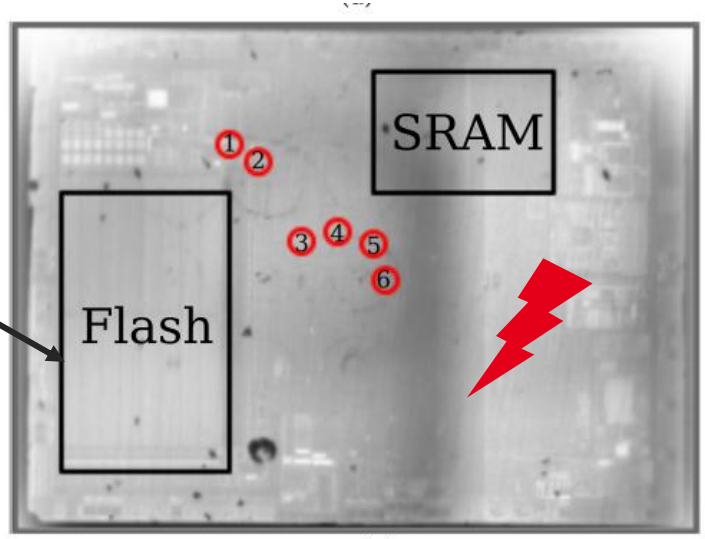
Discrimination: ability of a test to locate faults relying on their manifestation
Discrimination Rate (DR): number of faults leading to a fault manifestation



➔ **Objective: Ideally DR = 1**

Current approaches DR

```
i1:  add r0, r0, 0x1
i2:  add r0, r0, 0x2
i3:  add r0, r0, 0x3
i4:  add r0, r0, 0x4
i5:  add r1, r1, 0x1
i6:  add r2, r2, 0x1
i7:  add r3, r3, 0x1
i8:  add r4, r4, 0x1
i9:  add r0, r0, 0x5
i10: add r0, r0, 0x6
i11: add r0, r0, 0x7
i12: add r0, r0, 0x8
```



Extracted from [khuat, 2021]

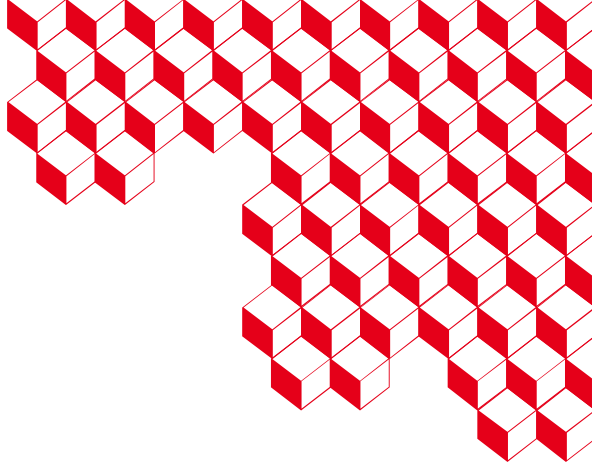
r2 not updated properly

➔ DR=24 (on CV32E40P RISCV CPU)

Why?

- Current approaches programs are handwritten
  → small set of programs to characterize a full CPU

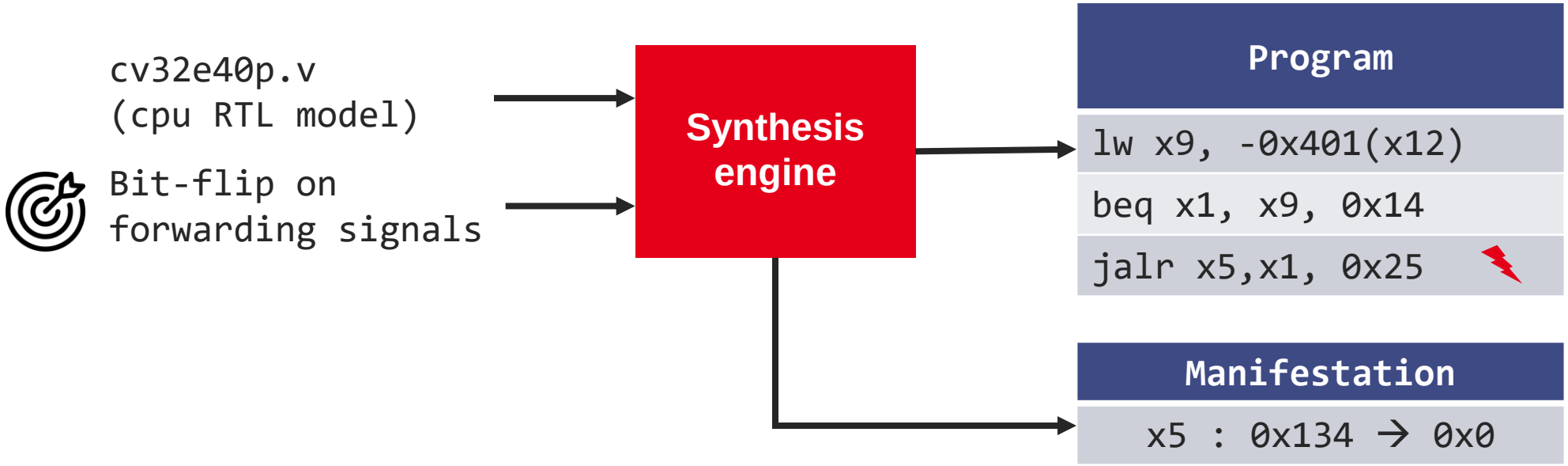
[khuat, 2021] - Vanthanh Khuat, Jean-Luc Danger, Jean-Max Dutertre. Laser Fault Injection in a 32-bit Microcontroller: from the Flash Interface to the Execution Pipeline. 2021 Workshop on Fault Detection and Tolerance in Cryptography (FDTC)



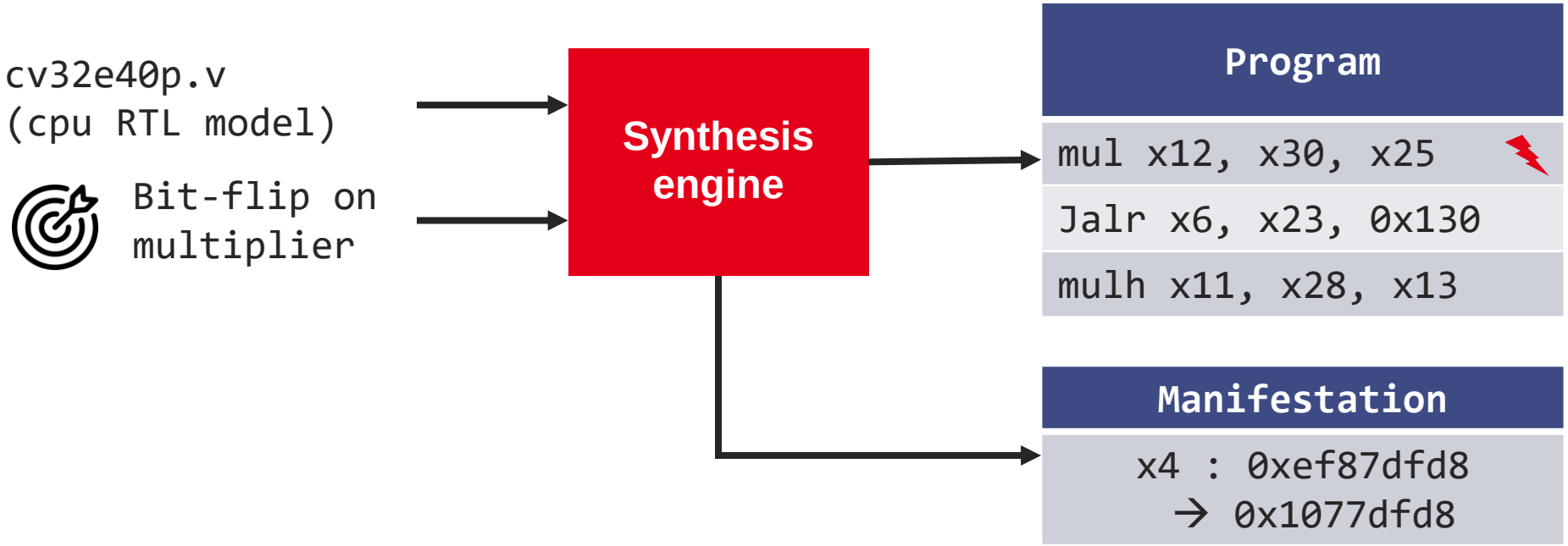
“How could we automate characterization programs creation and evaluate their discrimination?”

Proposal: Program synthesis using model checking

Bit-flip = fault changing a signal from **0 to 1** or **1 to 0,**
commonly observed with **laser fault injection**

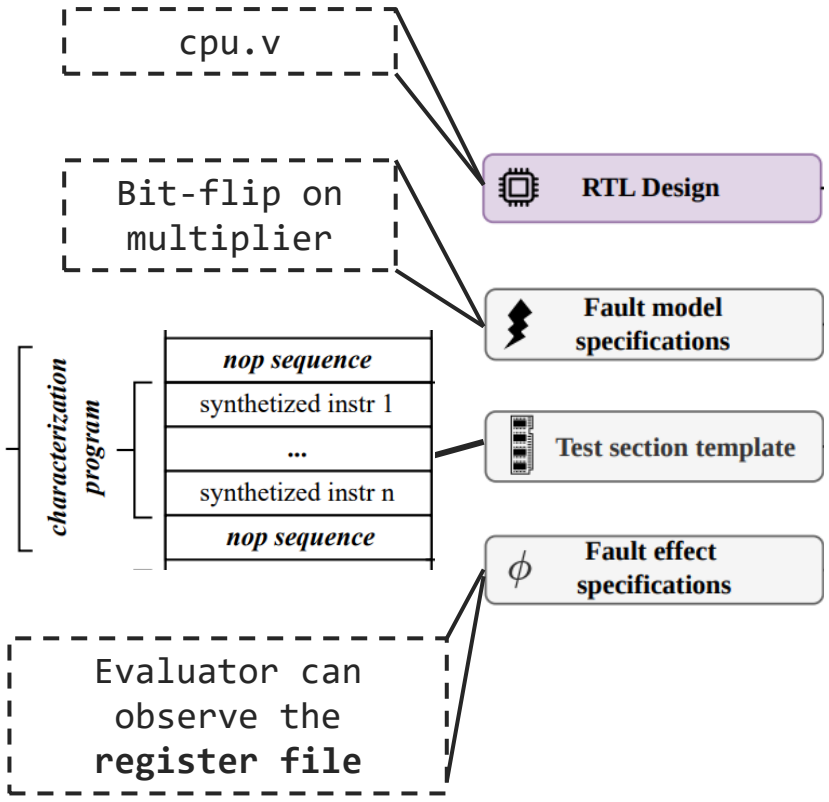


Proposal: Program synthesis using model checking

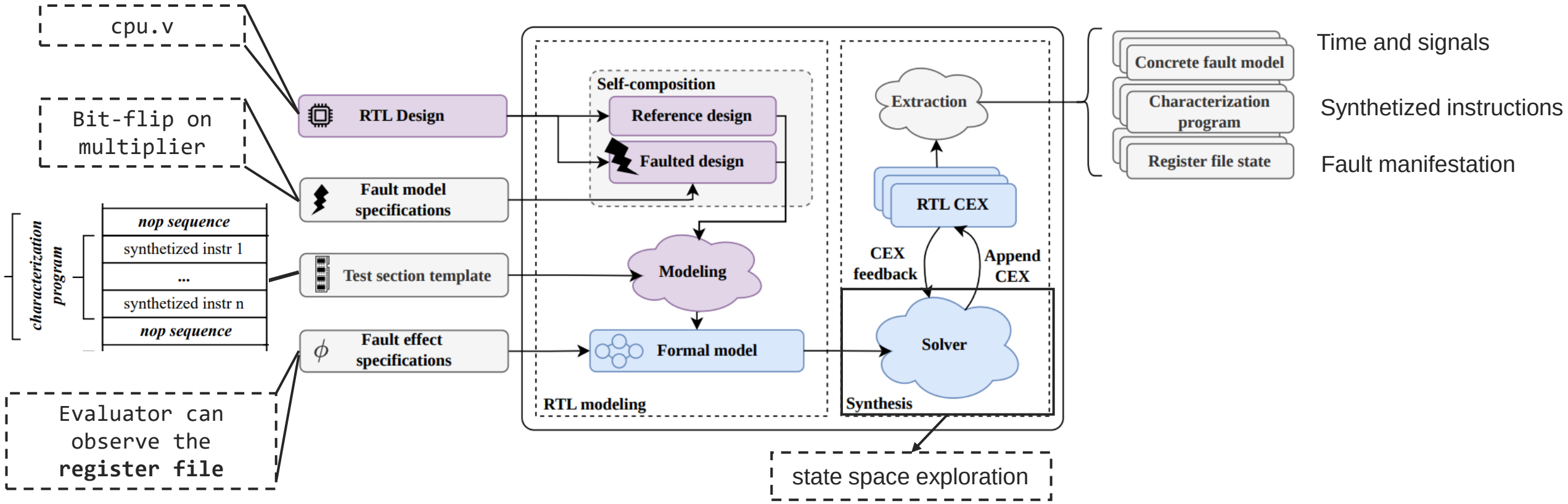


Methodology

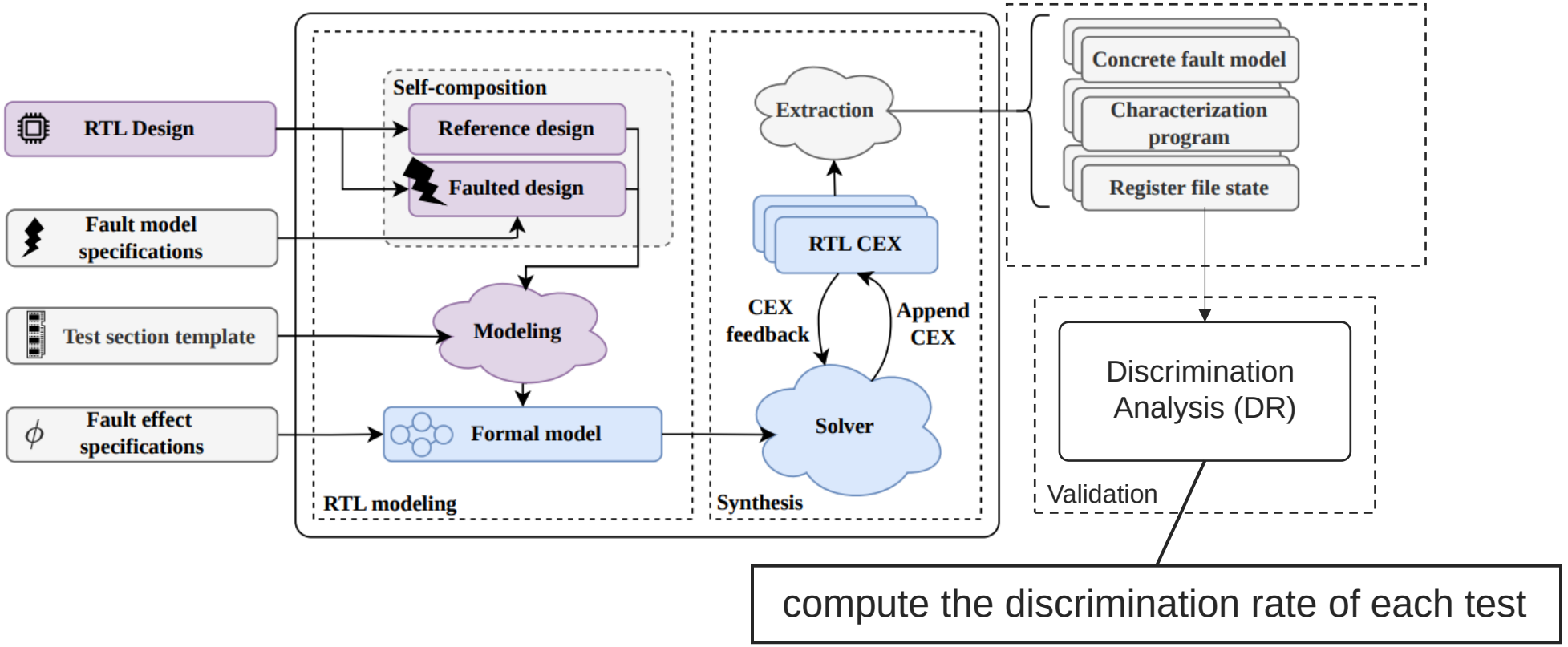
Methodology: RTL Modeling



Methodology: Synthesis



Methodology



Validation

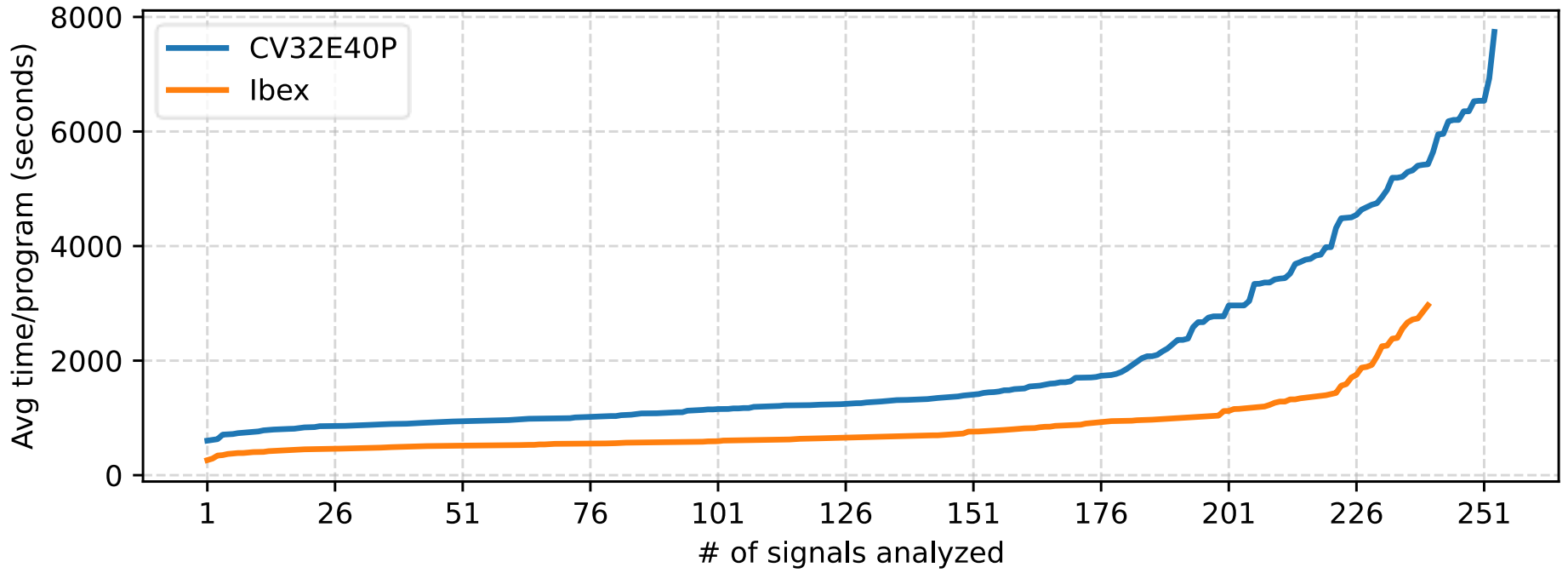
Validation: experimental setup

	CV32E40P	Ibex
Size (Yosys internal cells)	2498	1749
Architecture	RISC-V 32bits IMC	RISC-V 32bits IMC
Stages #	4	3
Experimental parameters		
Fault model	Bit flip	
Scope	Each control signals (size =< 2)	
Program length	3 instructions	
Methodology	Iterate synthesis over all signals in the scope	

→ Each signal has its characterization test

Validation: global figures

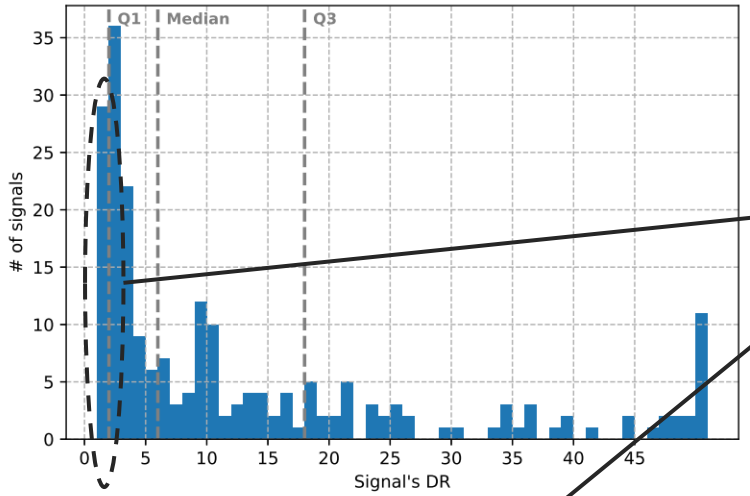
Results		
	CV32E40P	**Ibex**
Coverage	76%	64%
Median synthesis time	20mins50s	10mins42s
# of synthetized programs	1213	1135
Full analysis time	4.8 days	1.6 days



Validation: Program Discrimination

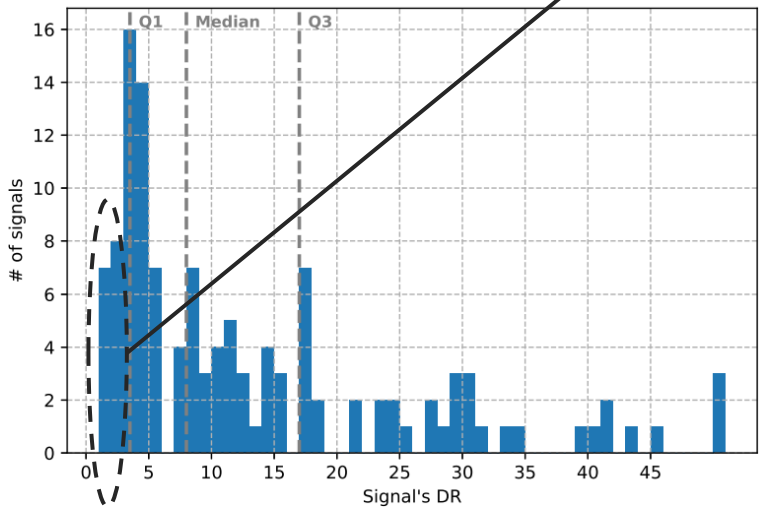
Number of signals revealed by its characterization test (smaller is better)

CV32E40P



Perfectly discriminant programs (DR=1)
CV32E40P: 29 signals
Ibex: 7 signals

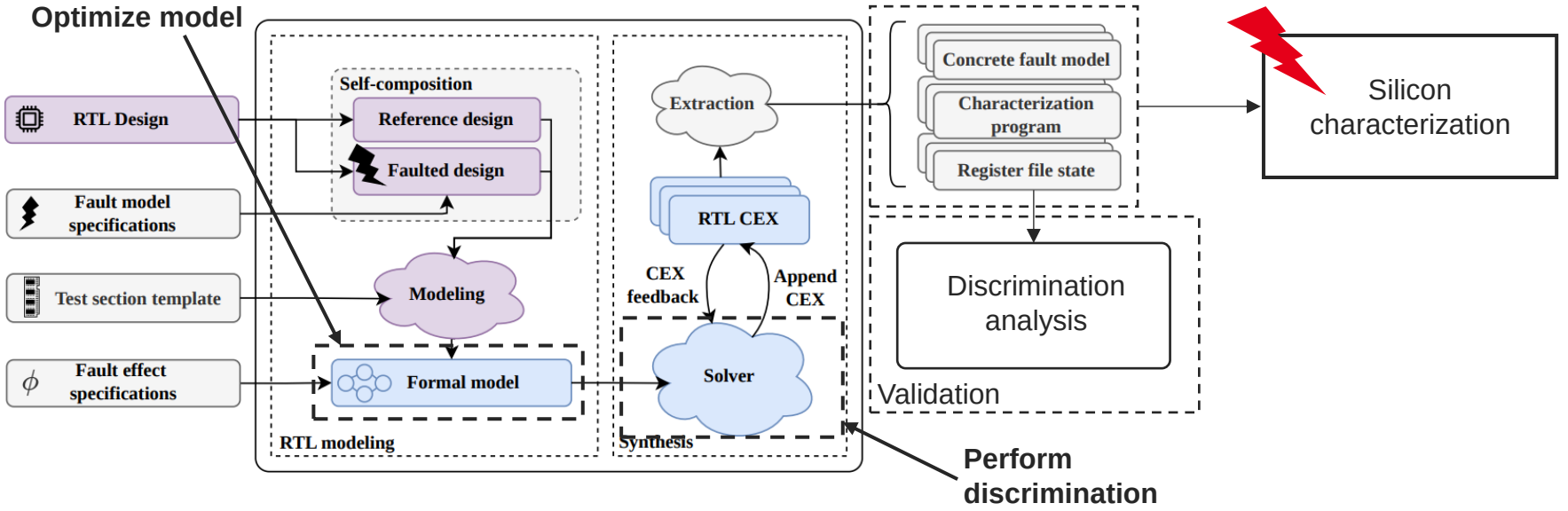
Ibex



Other programs
✅ Enumeration of signals → **Great asset for evaluators**
Median discrimination:
- **CV32E40P**: 6 signals
- **Ibex**: 7 signals

For more than 75% → DR < 24

Conclusion



Contributions summary:

→ An automated fault characterization program synthesis methodology

→ A discrimination analysis for characterization program

→ An evaluation over two open source test cases

Next steps:

→ Silicon characterization test

→ Time optimization: strategies to reduce synthesis time

→ Perform discrimination at synthesis time

Thank you

Signal proportion per size

TABLE II: Number of signals in $\mathcal{C}$ by width.

	Bit width	0:2	0:16	0:32	all
CV32E40P	# of signals	399	502	665	702
	% of design	56	71	94	100
Ibex	# of signals	475	551	721	786
	% of design	60	70	91	100

Coverage over larger signals

TABLE VII: Coverage for CV32E40P ($\phi^{7,10}_{\text{dump}}$), for signals with sizes ranging from 1 to 2, and 3 to 32 bits.

Module	Synthesis time	Coverage#	$\lvert\mathcal{C}_{\text{analyzed}}\rvert$
$\mathcal{C}[0:2]$	117h10m14	255	332
$\mathcal{C}[3:32]$	54h10m07	164	256

Impact of instruction length

l		1	2	3	4	5	6	7	8
Coverage#		8	10	10	10	10	10	10	10
Syn. time		658	888	818	873	1239	1986	1961	2646
DR	Q1	2	2	4	3	4	3	2	3
	med	7	4	9	4	8	7	11	3
	Q3	30	37	26	13	13	22	23	25