Lost in Translation: Enabling Confused Deputy Attacks on EDA Software with TransFuzz

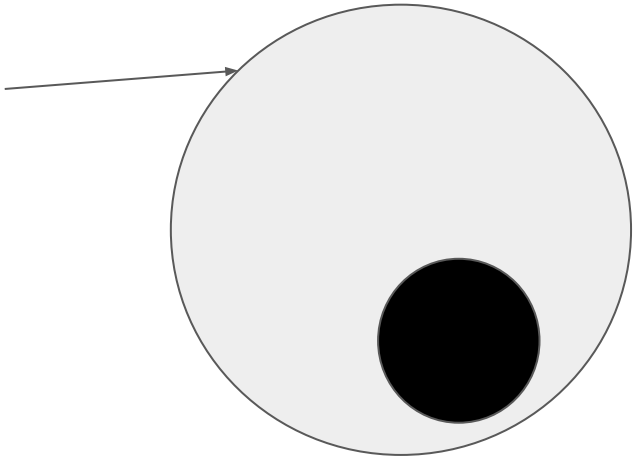
Flavien Solt, Kaveh Razavi
ETH Zurich

USENIX Security 2025

ETH zürich | COMSEC

Overview

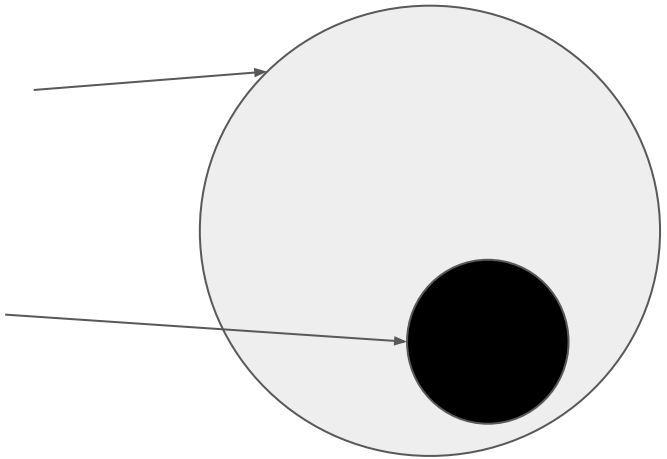
Functional verification: once a functionality is covered, then we are happy.



Overview

Functional verification: once a functionality is covered, then we are happy.

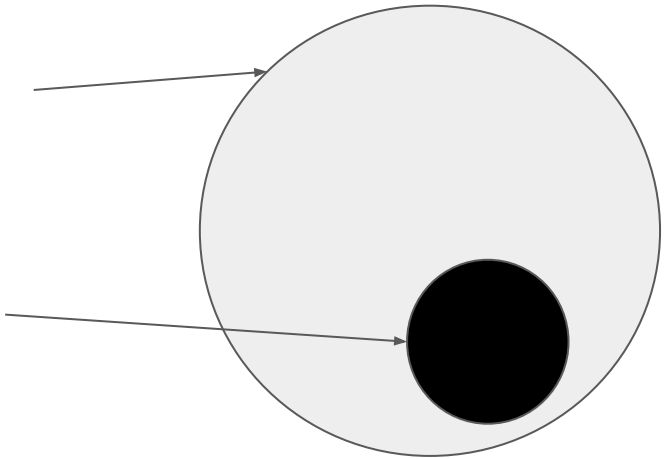
Unsoundness: We miss bugs that are in coverage scope.



Overview

Functional verification: once a functionality is covered, then we are happy.

Unsoundness: We miss bugs that are in coverage scope.

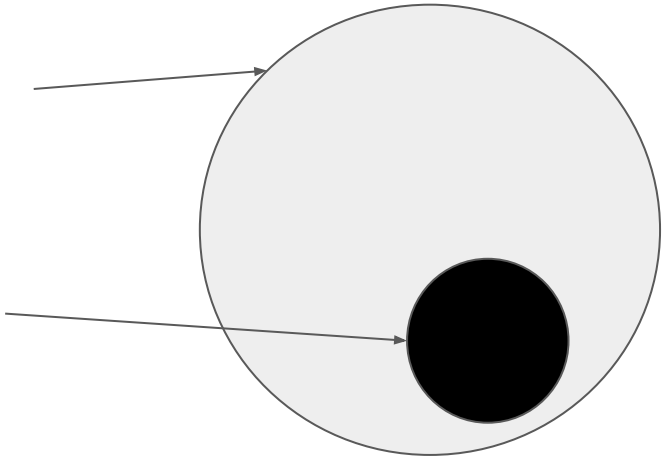


Because of bugs

Overview

Functional verification: once a functionality is covered, then we are happy.

Unsoundness: We miss bugs that are in coverage scope.



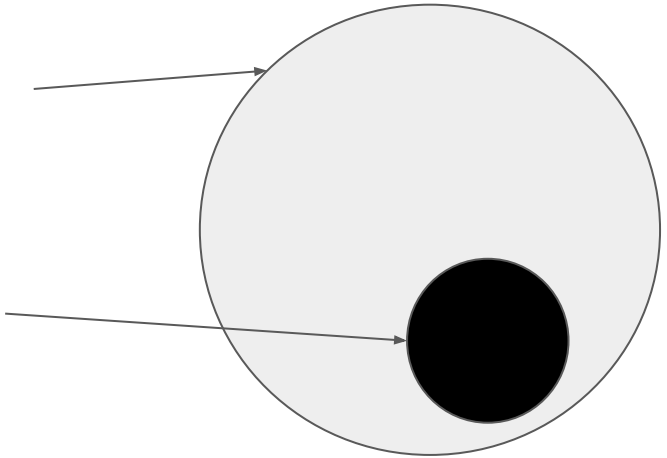
Because of bugs

Bugs in verification tools

Overview

Functional verification: once a functionality is covered, then we are happy.

Unsoundness: We miss bugs that are in coverage scope.



Because of bugs

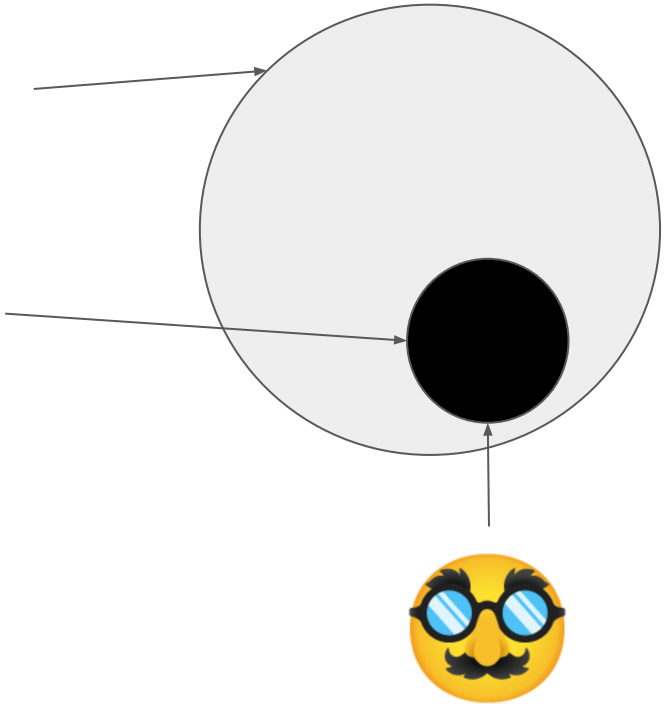
Bugs in verification tools

Bugs in EDA involved after verification: TOCTOU

Overview

Functional verification: once a functionality is covered, then we are happy.

Unsoundness: We miss bugs that are in coverage scope.



Because of bugs

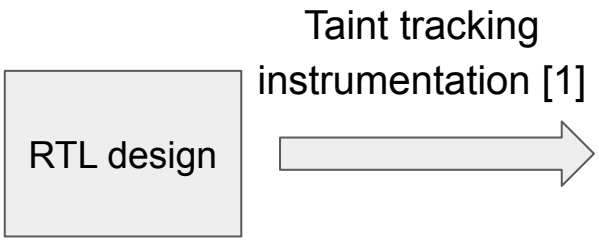
Bugs in verification tools

Bugs in EDA involved after verification: TOCTOU

Overview

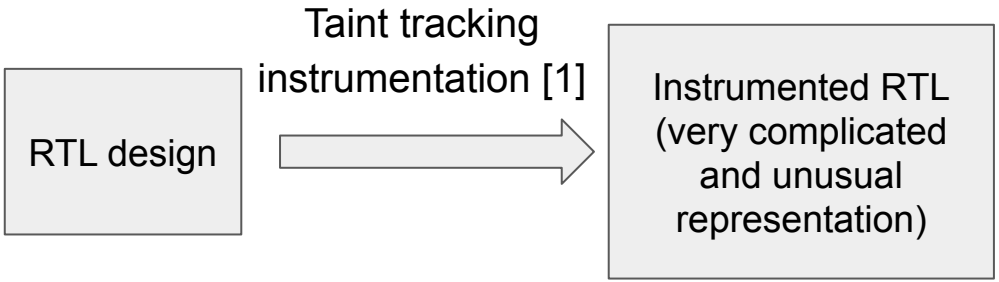
Why have we been looking at unsoundness?

Motivation 1: Errors due to instrumentation



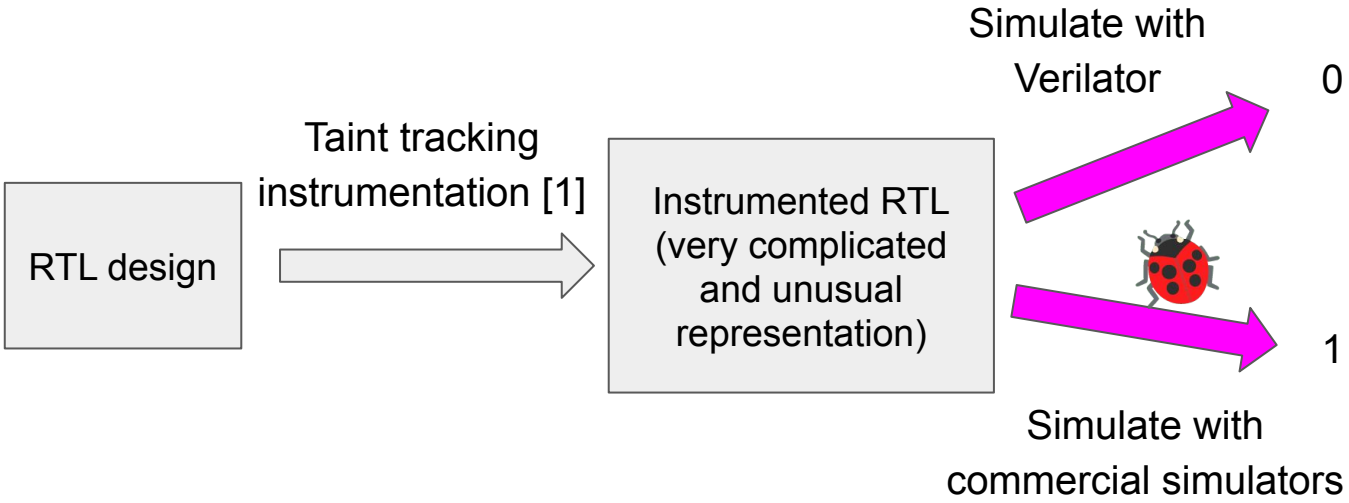
[1] F. Solt, B. Gras, K. Razavi, “CellIFT: Leveraging Cells for Scalable and Precise Dynamic Information Flow Tracking in RTL.”, USENIX Security 2022

Motivation 1: Errors due to instrumentation



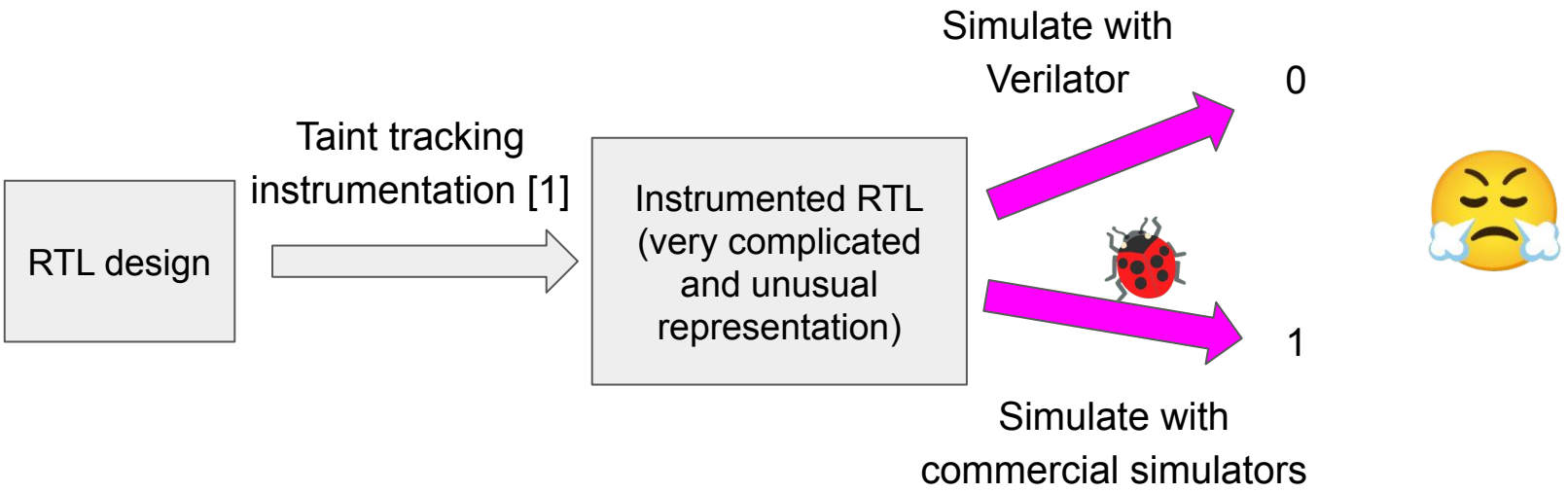
[1] F. Solt, B. Gras, K. Razavi, “CellIFT: Leveraging Cells for Scalable and Precise Dynamic Information Flow Tracking in RTL.”, USENIX Security 2022

Motivation 1: Errors due to instrumentation



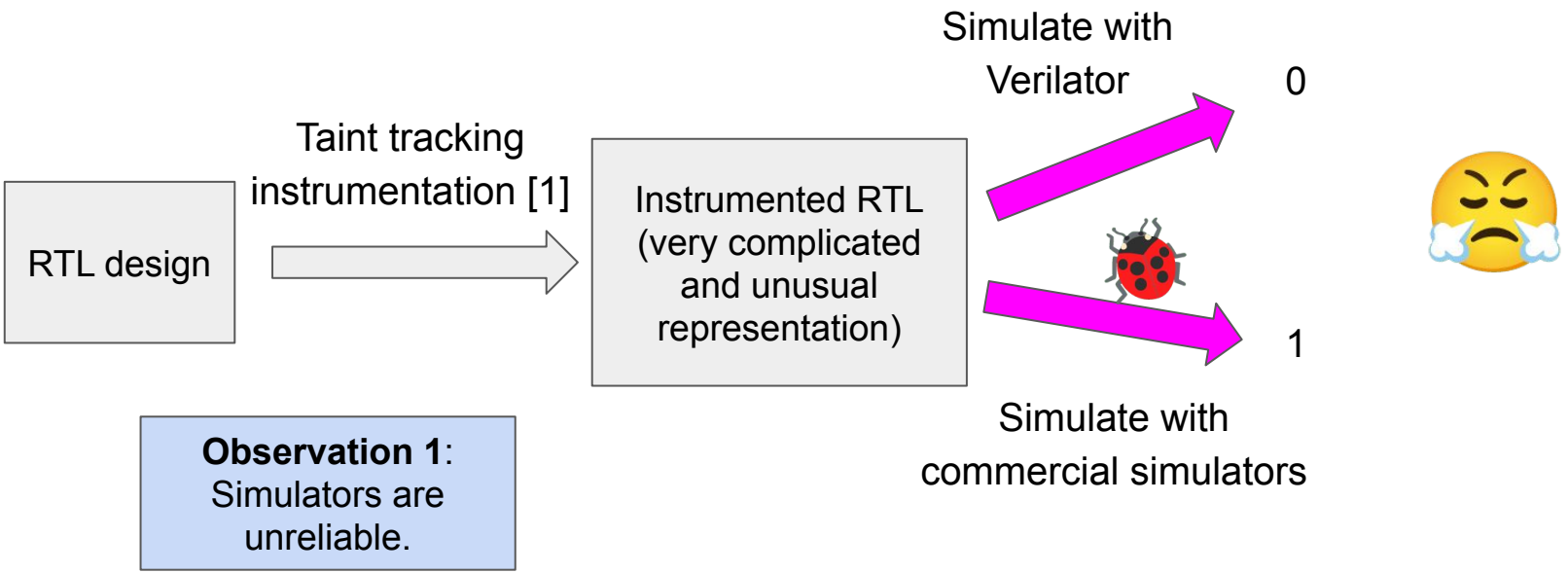
[1] F. Solt, B. Gras, K. Razavi, “CellIFT: Leveraging Cells for Scalable and Precise Dynamic Information Flow Tracking in RTL.”, USENIX Security 2022

Motivation 1: Errors due to instrumentation



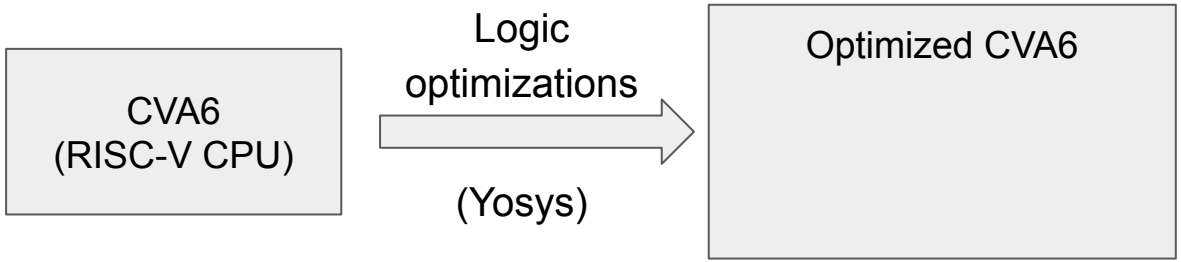
[1] F. Solt, B. Gras, K. Razavi, “CellIFT: Leveraging Cells for Scalable and Precise Dynamic Information Flow Tracking in RTL.”, USENIX Security 2022

Motivation 1: Errors due to instrumentation



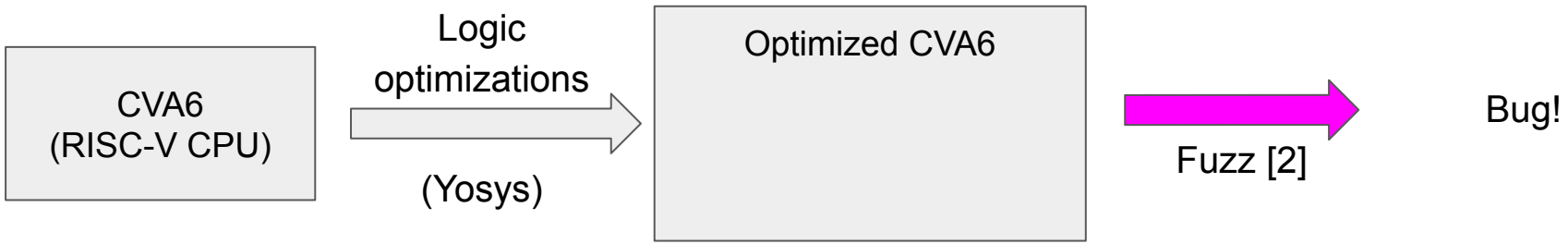
[1] F. Solt, B. Gras, K. Razavi, “CellIFT: Leveraging Cells for Scalable and Precise Dynamic Information Flow Tracking in RTL.”, USENIX Security 2022

Motivation 2: Errors due to optimizations



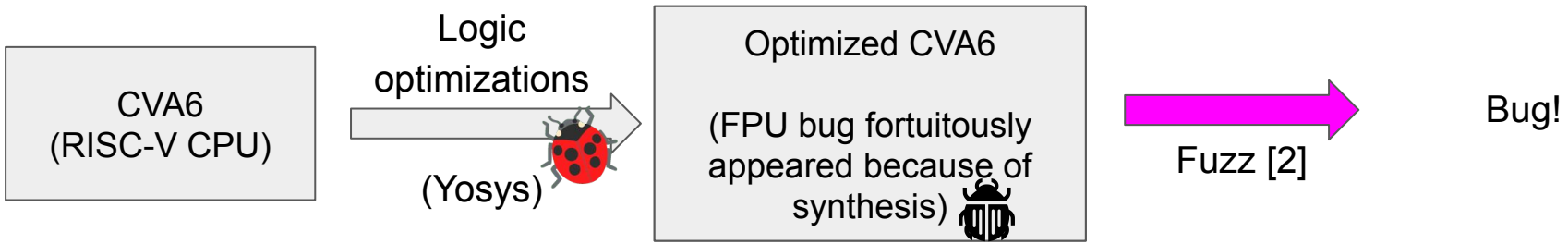
[2] F. Solt, K. Ceesay-Seitz, K. Razavi, “Cascade: CPU Fuzzing via Intricate Program Generation.”, USENIX Security 2024

Motivation 2: Errors due to optimizations



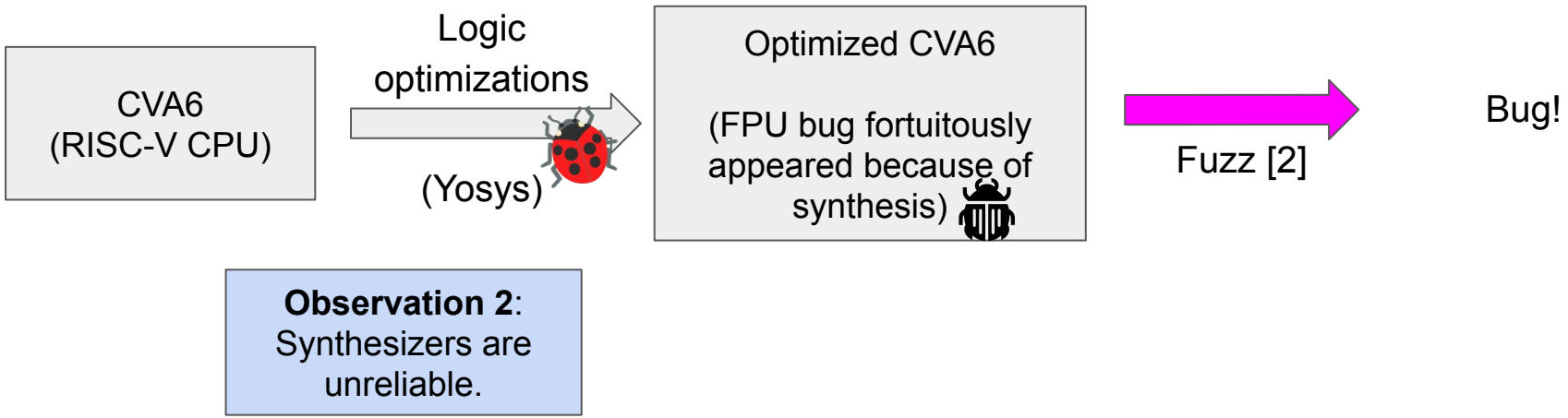
[2] F. Solt, K. Ceesay-Seitz, K. Razavi, “Cascade: CPU Fuzzing via Intricate Program Generation.”, USENIX Security 2024

Motivation 2: Errors due to optimizations



[2] F. Solt, K. Ceesay-Seitz, K. Razavi, “Cascade: CPU Fuzzing via Intricate Program Generation.”, USENIX Security 2024

Motivation 2: Errors due to optimizations



[2] F. Solt, K. Ceesay-Seitz, K. Razavi, “Cascade: CPU Fuzzing via Intricate Program Generation.”, USENIX Security 2024

What can a malicious person do?

Threat models:

- Either a malicious contributor to an open-source repository.
- Or a malicious external IP provider.

What can a malicious person do?

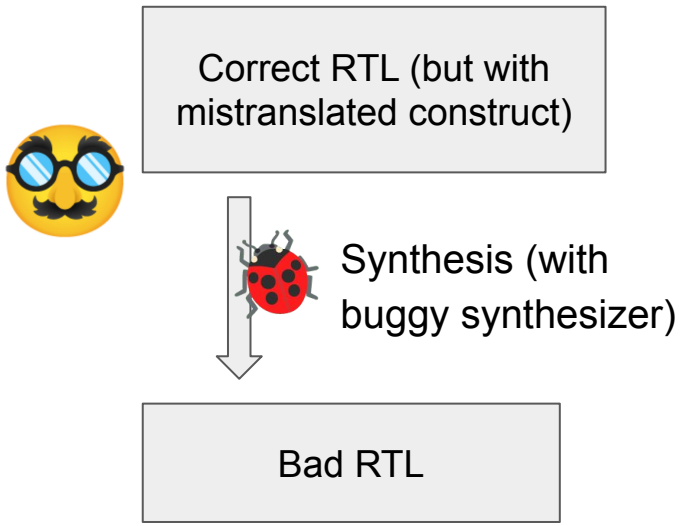
Assume a
synthesizer
translation bug

What can a malicious person do?

Correct RTL (but with mistranslated construct)

Assume a **synthesizer** translation bug

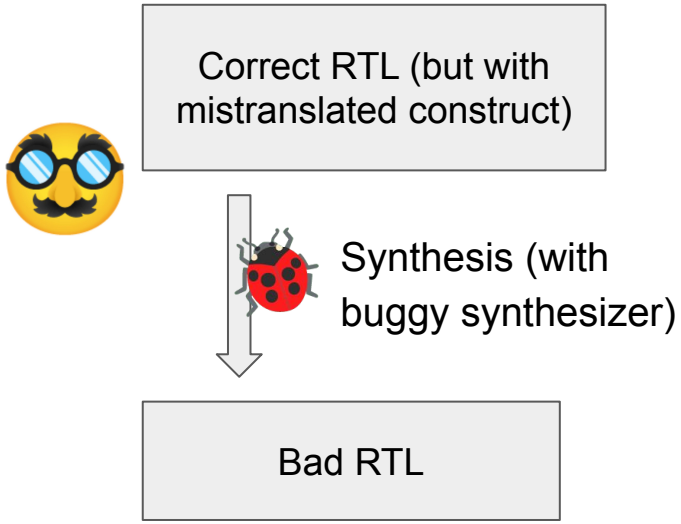
What can a malicious person do?



This is a TOCTOU attack.

Assume a **synthesizer** translation bug

What can a malicious person do?

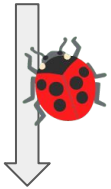


Assume a **synthesizer** translation bug

Assume a **simulator/verifier** translation bug

What can a malicious person do?

Correct RTL (but with mistranslated construct)



Synthesis (with buggy synthesizer)

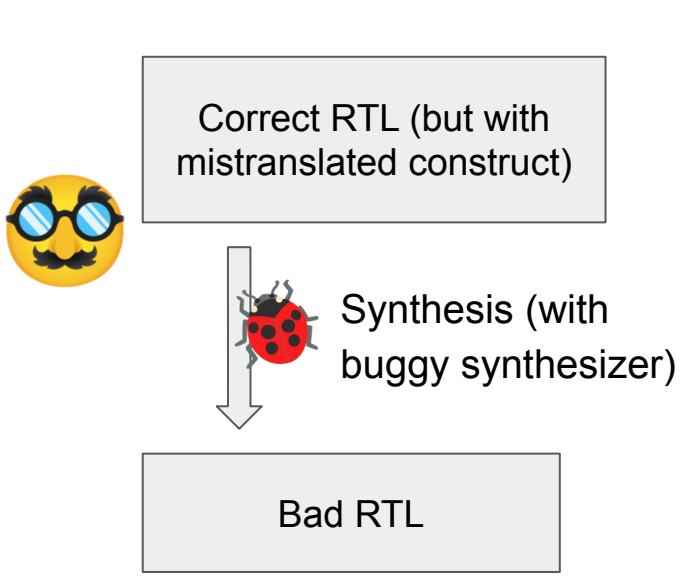
Bad RTL

Assume a **synthesizer** translation bug

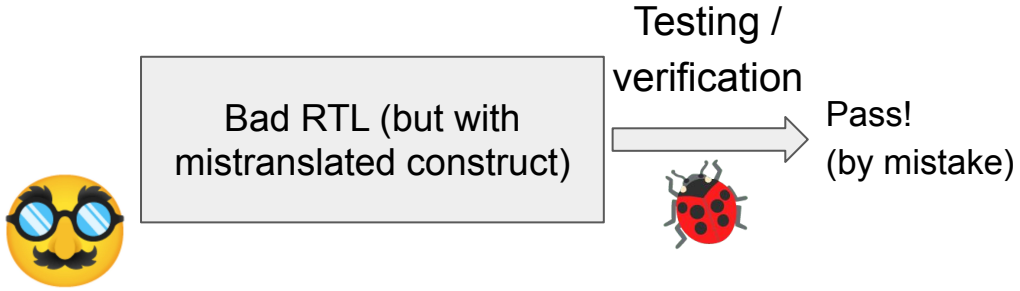
Bad RTL (but with mistranslated construct)

Assume a **simulator/verifier** translation bug

What can a malicious person do?

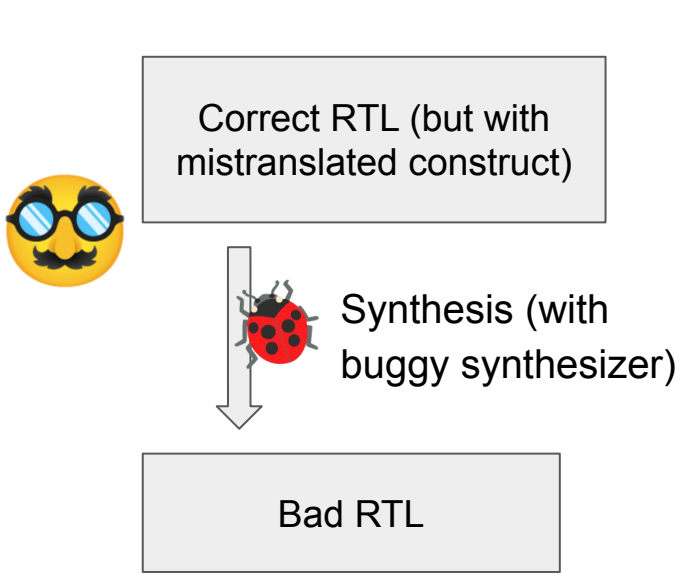


Assume a **synthesizer** translation bug

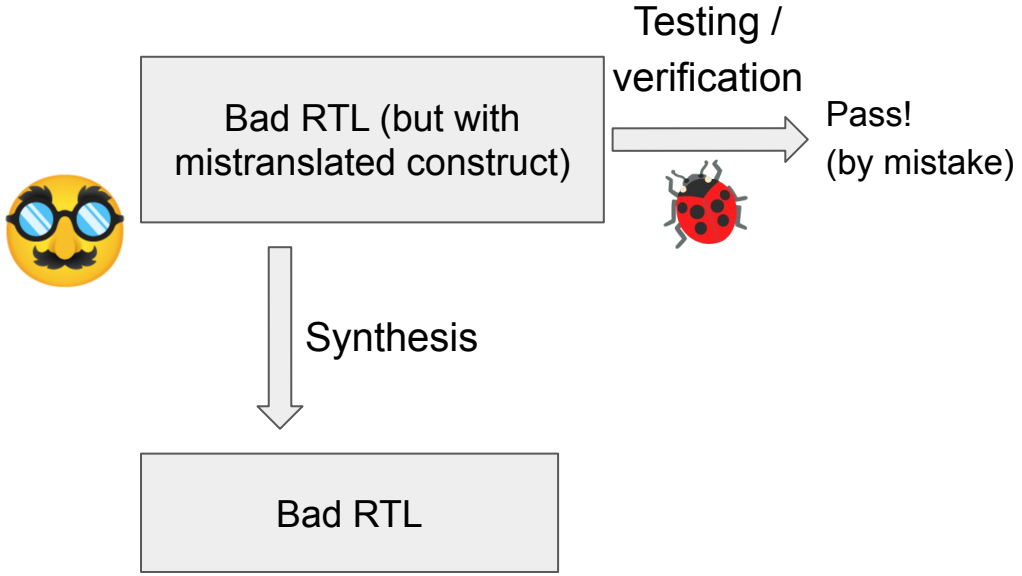


Assume a **simulator/verifier** translation bug

What can a malicious person do?



Assume a **synthesizer** translation bug



Assume a **simulator/verifier** translation bug

Requirements

 =

Requirements

TransFuzz:
differential fuzzing

1. Translation bugs.

=

Requirements

TransFuzz:
differential fuzzing

1. Translation bugs.

 = +

Mistranslation gadgets

2. Reliable bug packaging.

Requirements

=

1. Translation bugs.

+

2. Reliable bug packaging.

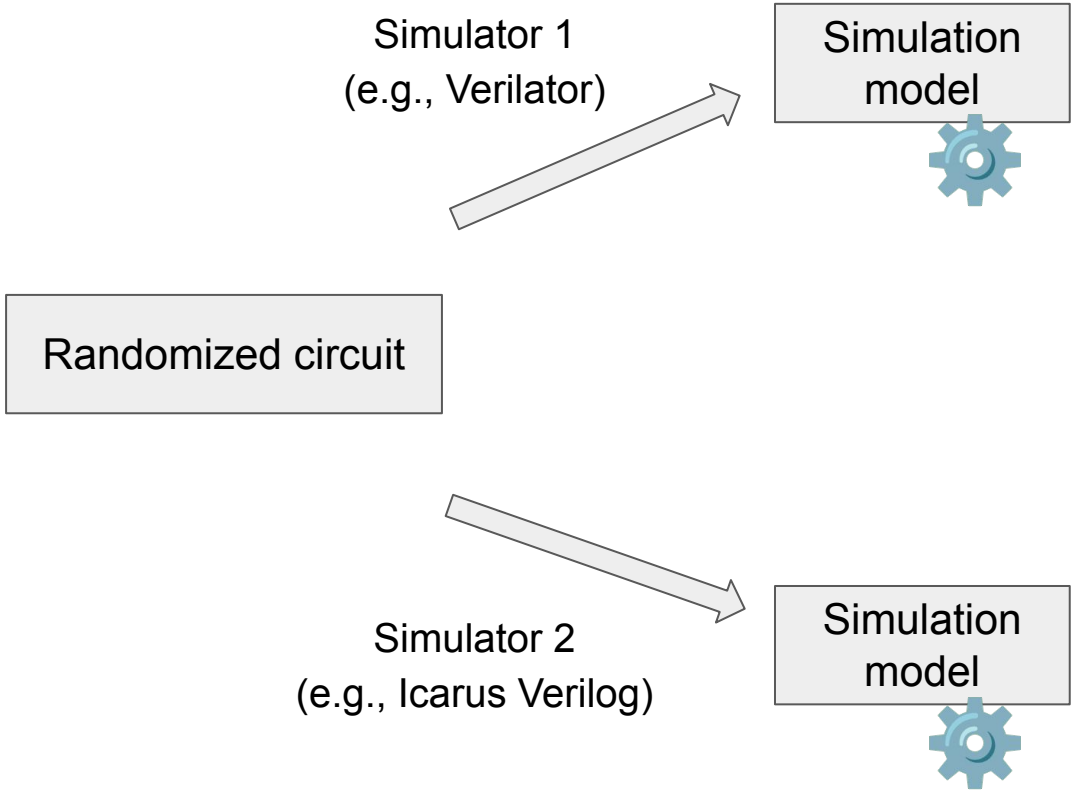
TransFuzz:
differential fuzzing

Mistranslation gadgets

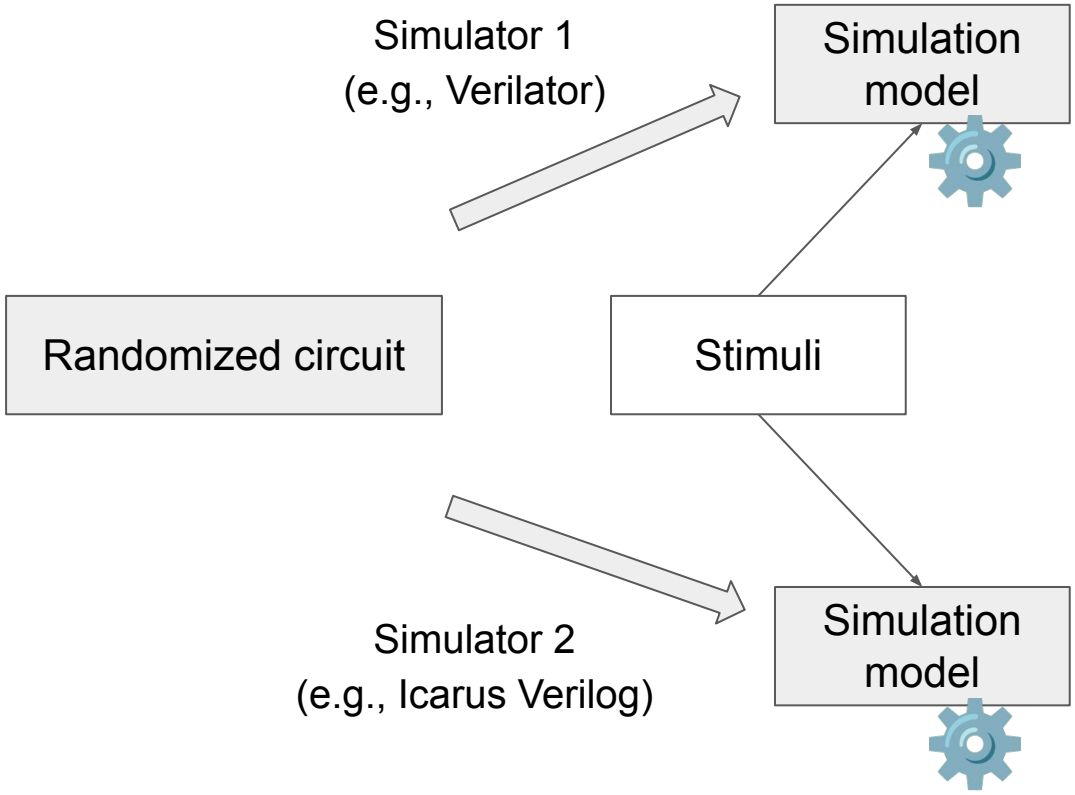
TransFuzz: finding translation bugs

Randomized circuit

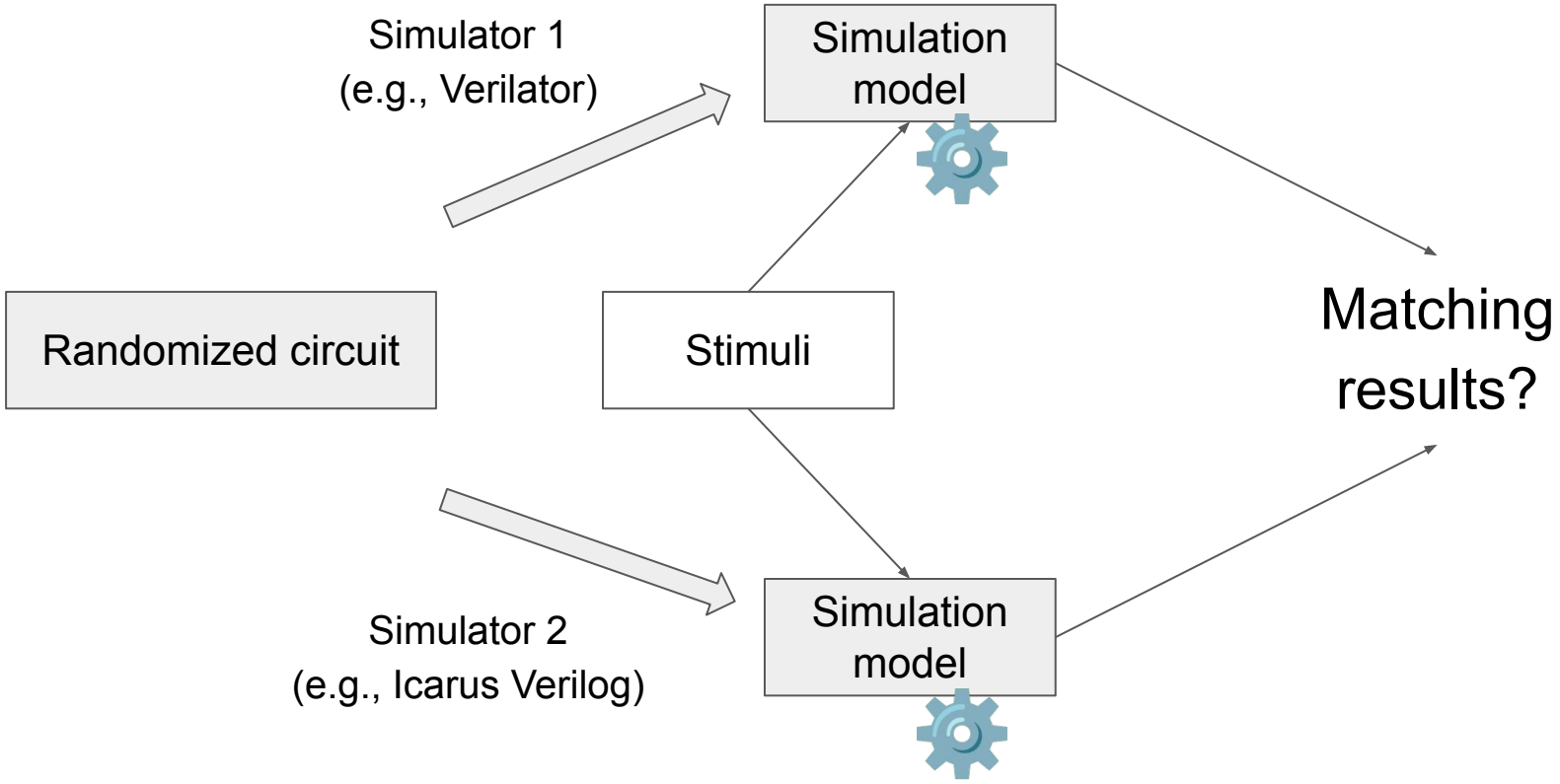
TransFuzz: finding translation bugs



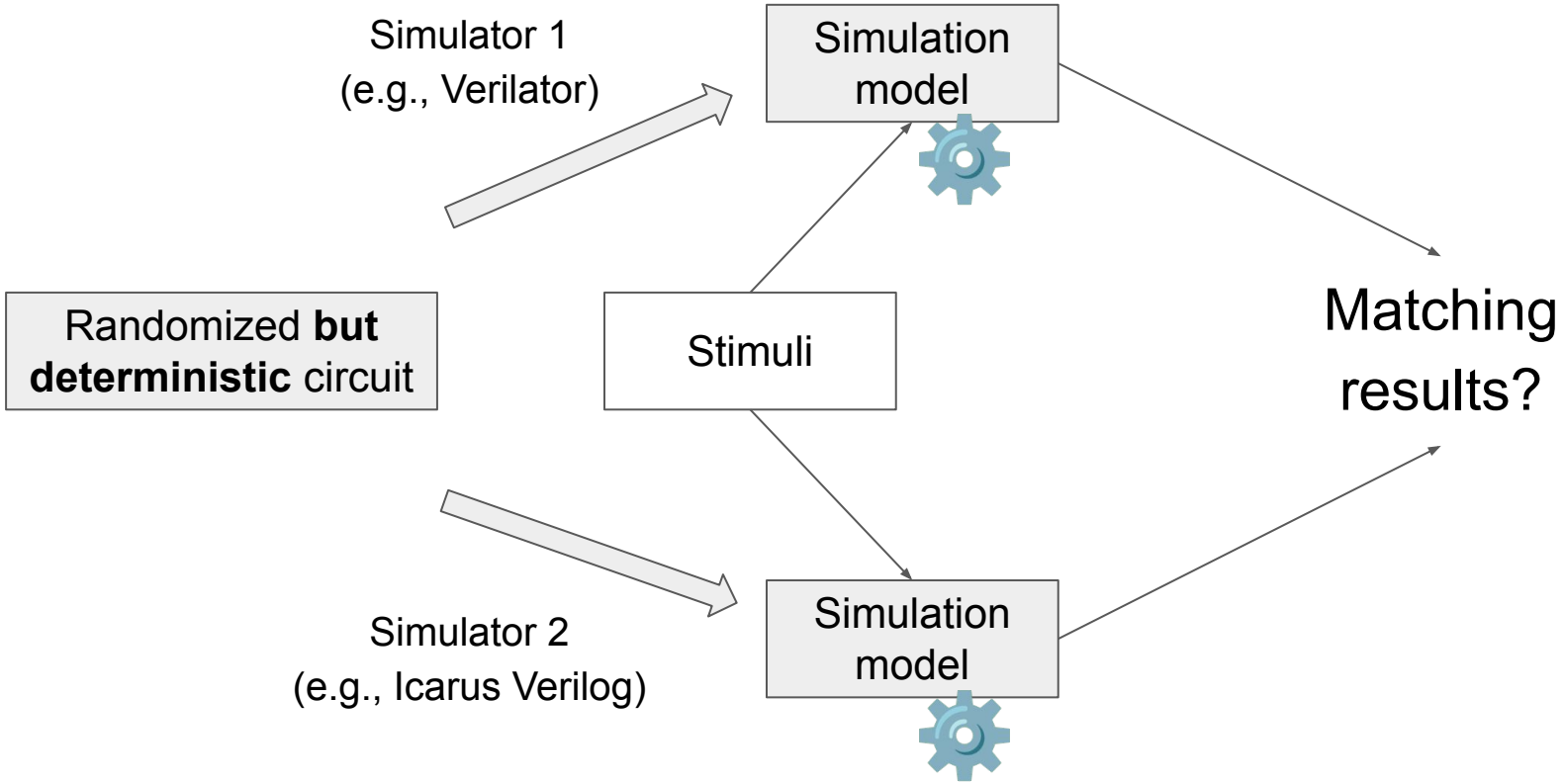
TransFuzz: finding translation bugs



TransFuzz: finding translation bugs



TransFuzz: finding translation bugs



25 CVEs
20 translation bugs

Id	Bug Description
I1	Segfault when using out-of-scope variable in slices
I2	Arithmetics deviation in specific circumstances
I3	Performance issue for a design with xor reductions
C1	Shifts consider signed operand in some cases
C2	Bad assignment under specific conditions
C3	Bug with modulo when operands intersect
C4	Both left shifts sometimes overflow the output signal
C5	Incorrect division
C6	Clock edges sometimes require 2 evaluations
C7	Performance issue with many concatenations
C8	Elaboration fails for some stateful cells in some cases
V1	Evasive malloc failure in some instances
V2	Segfault during evaluation

...

V17	Compiler sees empty input due to file system races
Y1	`opt_muxtree` wrong if twice the same mux
Y2	Misoptimization of wide shifts

I: Icarus Verilog
C: CXXRTL
V: Verilator
Y: Yosys

Requirements

=

1. Translation bugs.

+

2. Reliable bug packaging.

TransFuzz:
differential fuzzing

Mistranslation gadgets

Requirements

=

1. Translation bugs.

+

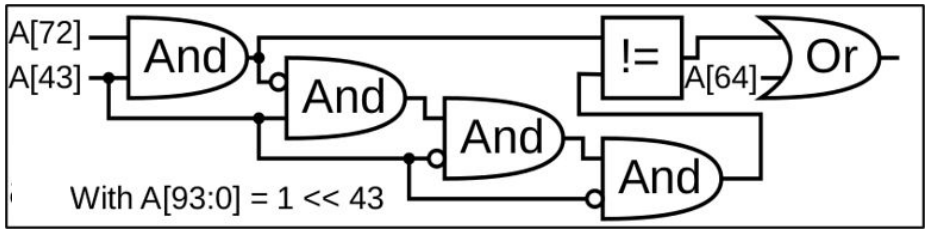
2. Reliable bug packaging.

TransFuzz:
differential fuzzing

Mistranslation gadgets

Gadgets

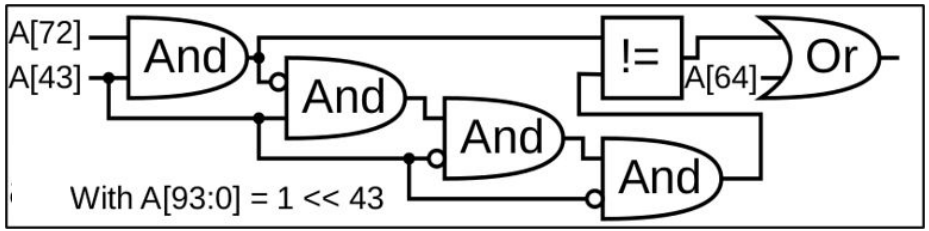
Should
produce 0



Gadgets

Should produce 0

Produces 1 because of the bug



Example gadget using a Verilator bug

Requirements

=

1. Translation bugs.

+

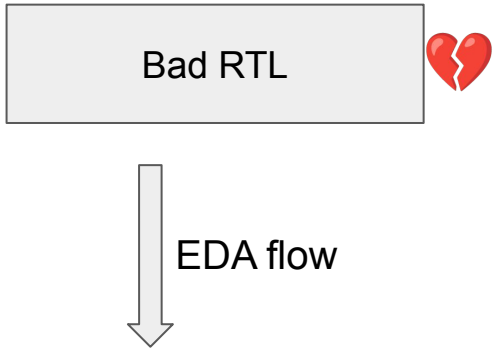
2. Reliable bug packaging.

✅ TransFuzz: differential fuzzing

✅ Mistranslation gadgets

Summary

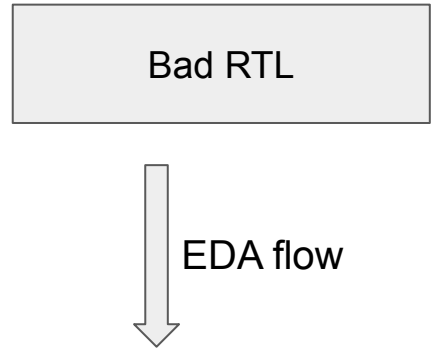
White-box techniques are an **effective** defense



Only dynamic testing is possible

Traditional setting

The backdoor is **undetectable**



Only dynamic testing is possible

Leveraging **translation bugs**

Proofs of concept:

1. Mistranslation gadget in the codebase
2. Mistranslation gadget in an external IP

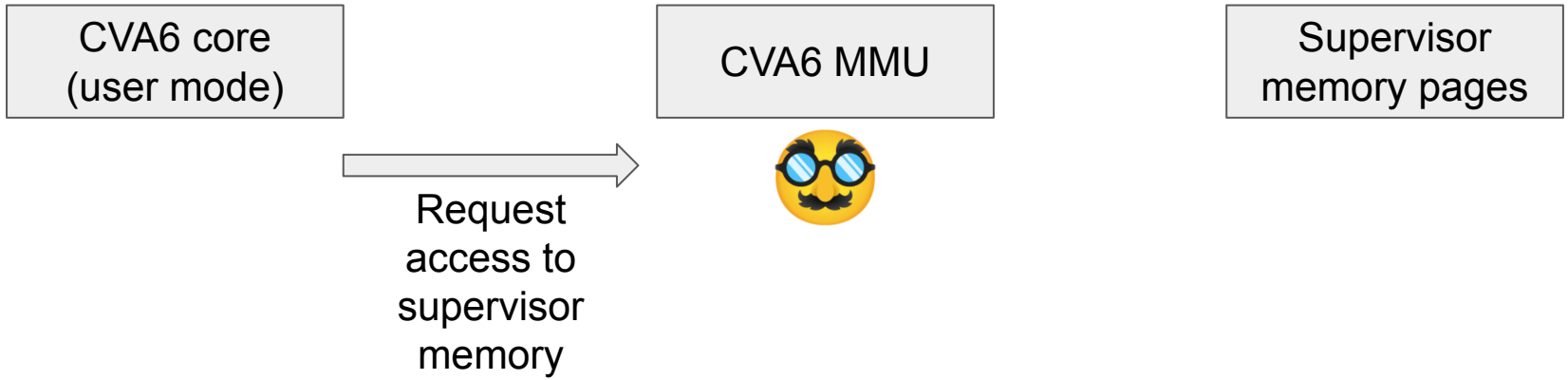
Proof of concept 1: MiRTL gadget in codebase

CVA6 core
(user mode)

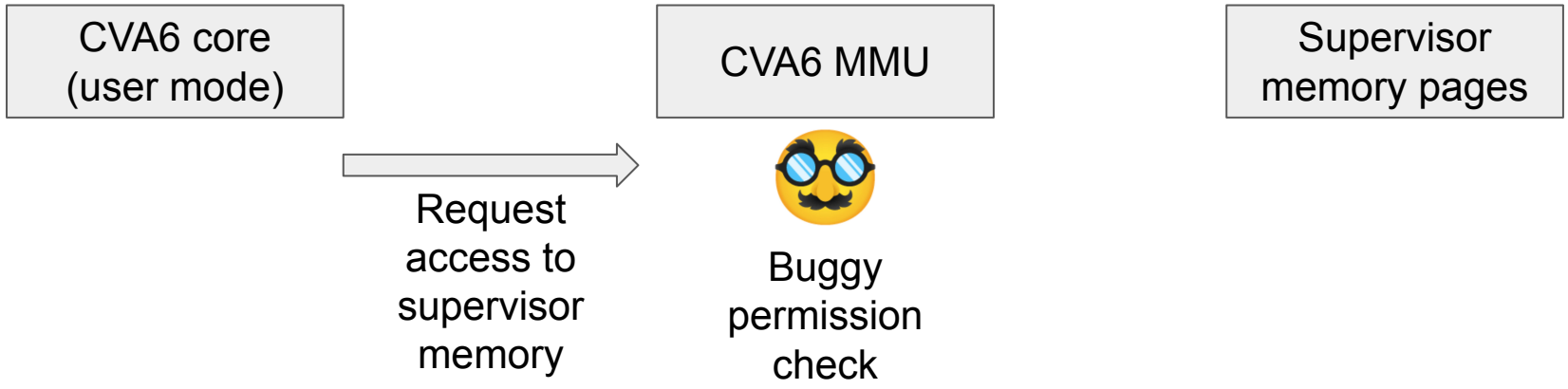
CVA6 MMU

Supervisor
memory pages

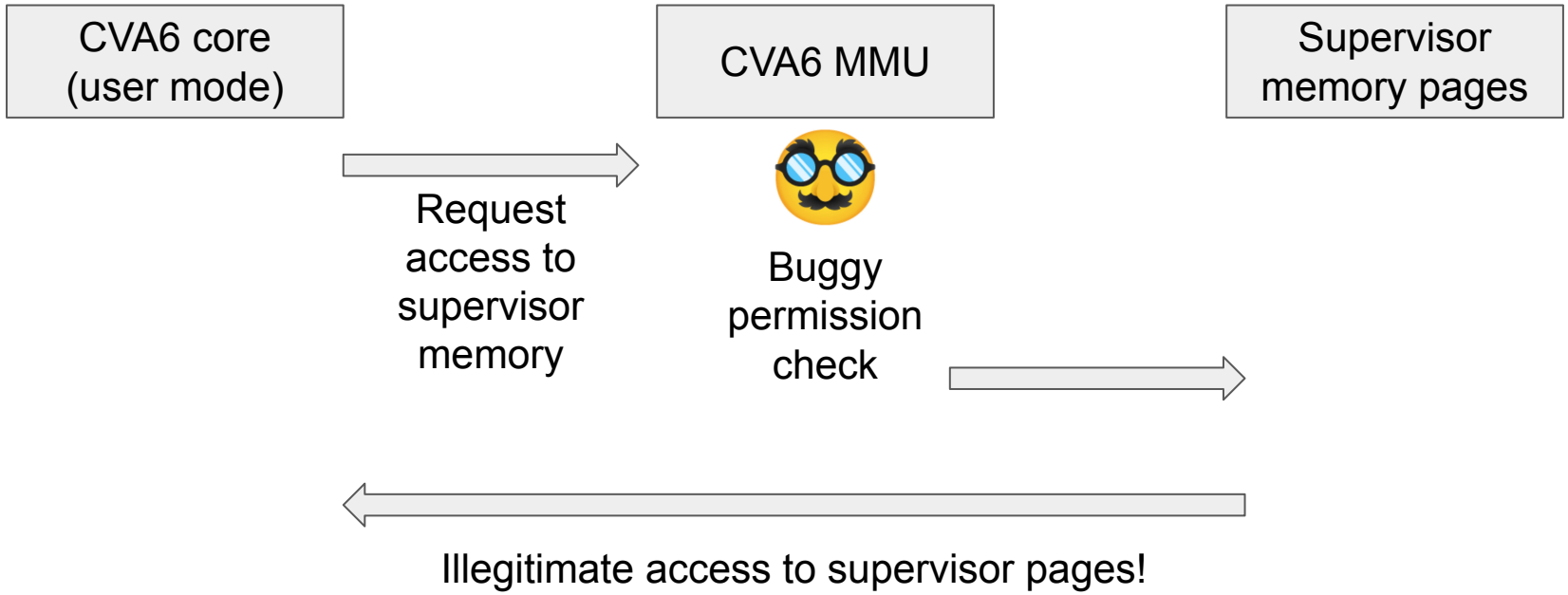
Proof of concept 1: MiRTL gadget in codebase



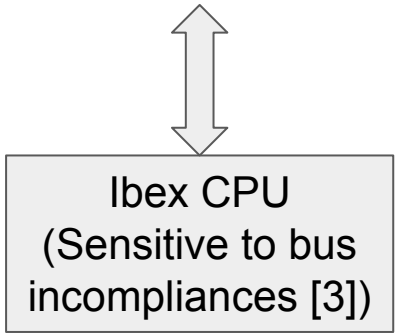
Proof of concept 1: MiRTL gadget in codebase



Proof of concept 1: MiRTL gadget in codebase

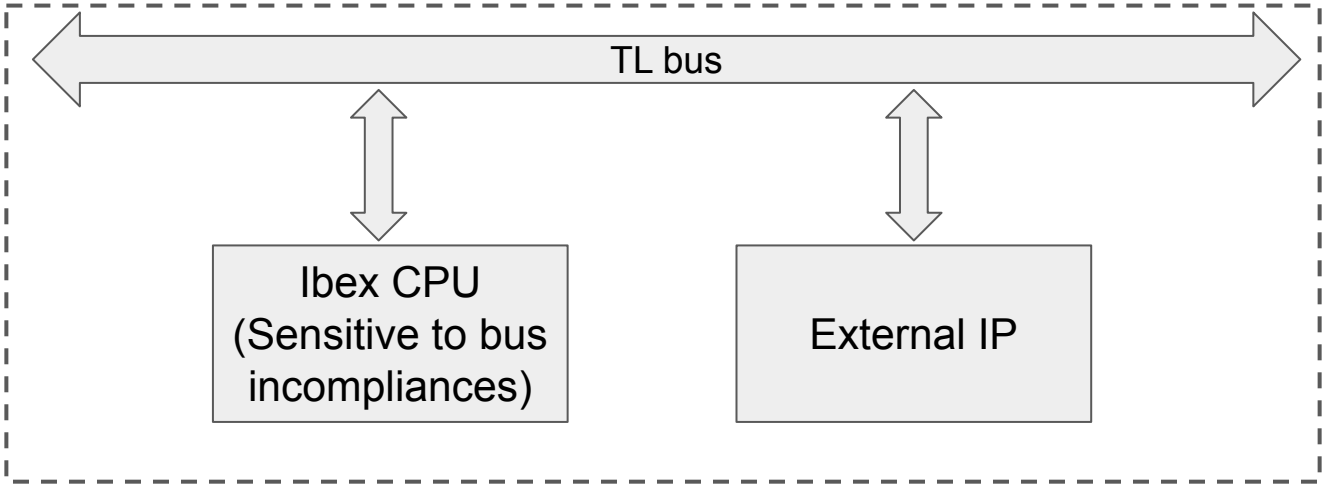


Proof of concept 2: MiRTL gadget in IP

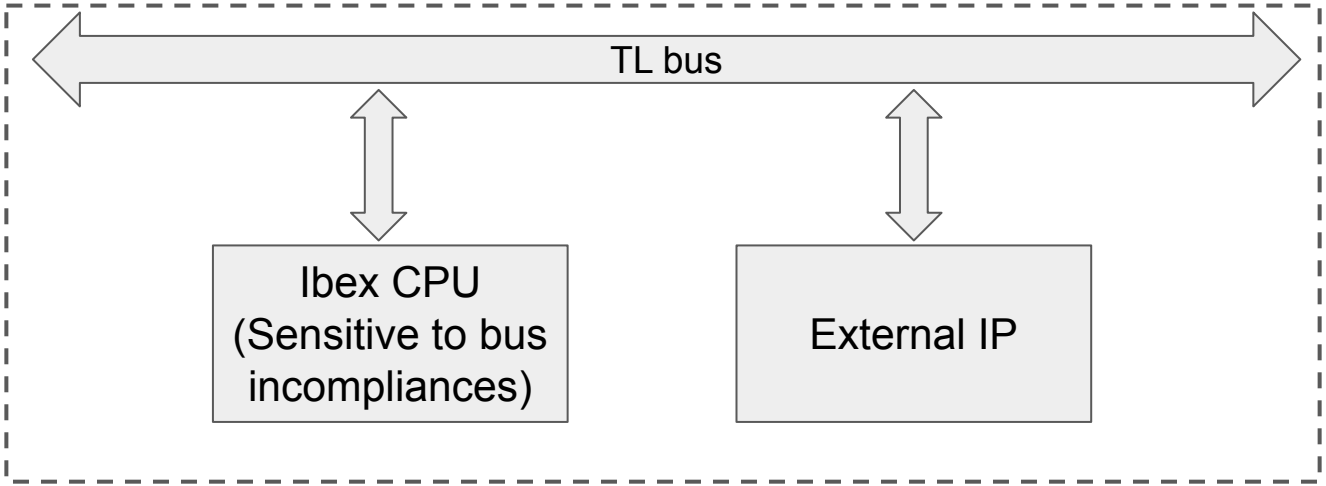


[3] K. Ceesay-Seitz, F. Solt, K. Razavi, “μCFI: Formal Verification of Microarchitectural Control-flow Integrity.”, CCS 2024

Proof of concept 2: MiRTL gadget in IP

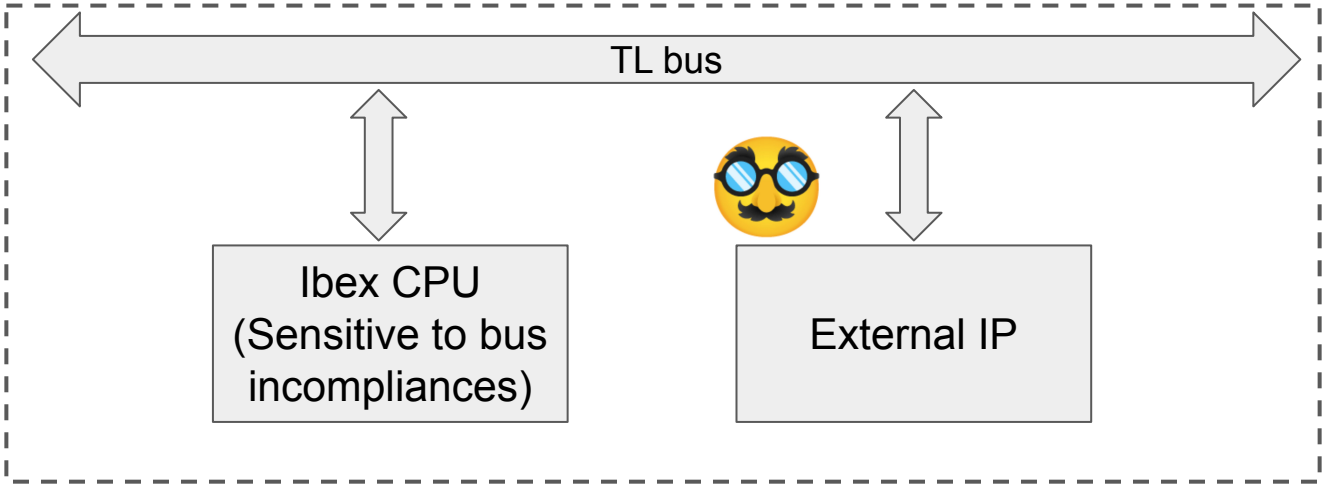


Proof of concept 2: MiRTL gadget in IP

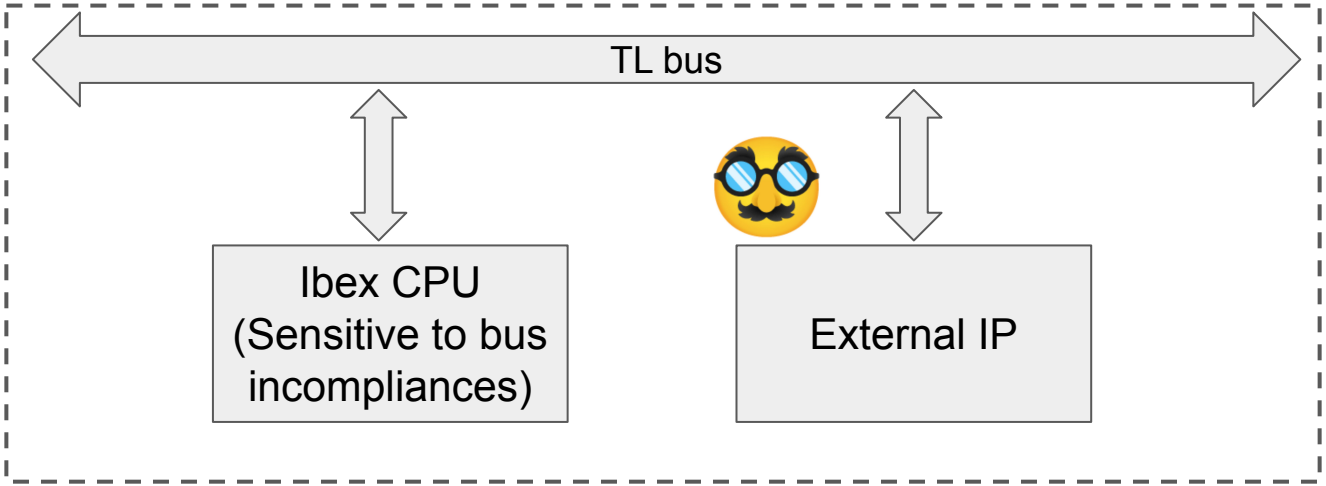


✅ Verify: no vulnerabilities!

Proof of concept 2: MiRTL gadget in IP

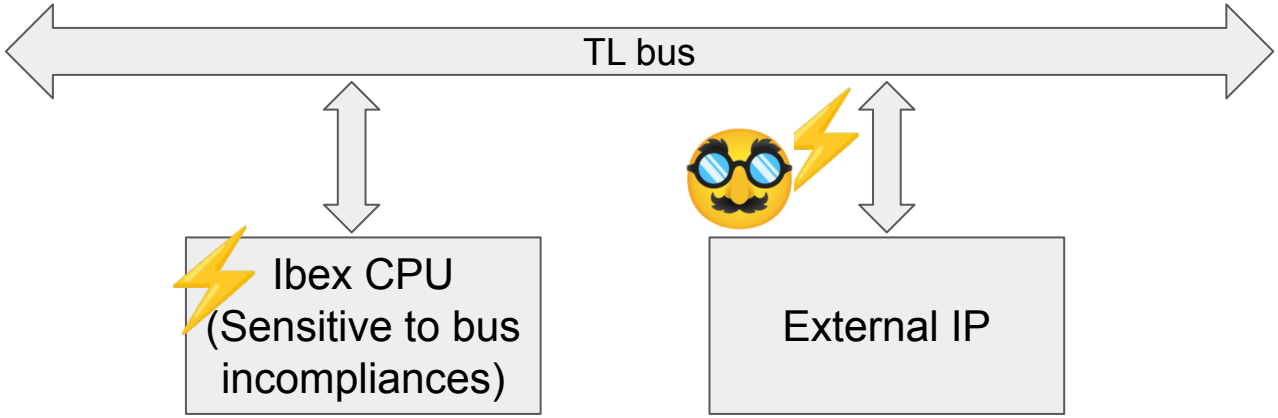


Proof of concept 2: MiRTL gadget in IP



✅ Verify: no vulnerabilities!

Proof of concept 2: MiRTL gadget in IP



Vulnerability enabled!

Conclusion

- Synthesizers and simulators are very buggy (25 CVEs).
- Bugs are easy to package into mistranslation gadgets.
- Mistranslation gadgets are nearly undetectable.
- We are pursuing defense work.

flavien.solt@berkeley.edu

flaviens