MorFuzz: Fuzzing Processor via Runtime Instruction Morphing enhanced Synchronizable Co-simulation

Jinyan Xu[1], Yiyuan Liu[1], Sirui He[2], Haoran Lin[1],
Yajin Zhou[1], and Cong Wang[2]

[1] Zhejiang University
[2] City University of Hong Kong

August, 2023

香港城市大學
City University of Hong Kong

Motivation

- Even the most advanced commercial processor is not perfect

Product	Last Update	#Errata	#Fixed
Intel 13th Generation	2023	42	2
Intel 12th Generation	2023	56	10
Intel 11th Generation	2023	36	10
Intel 10th Generation	2023	85+145	27+12
AMD EPYC 9004 Series	2023	40	0
AMD EPYC 7002 Series	2023	39	0
AMD EPYC 7003 Series	2022	62	0
AMD 1st/2nd Ryzen Series	2019	45/60	0

Motivation

- Even the most advanced commerc

Product	Last Update
Intel 13th Generation	2023
Intel 12th Generation	2023
Intel 11th Generation	2023
Intel 10th Generation	2023
AMD EPYC 9004 Series	2023
AMD EPYC 7002 Series	2023
AMD EPYC 7003 Series	2022
AMD 1st/2nd Ryzen Series	2019

r/sysadmin • 2 mo. ago
by acid_migrain

Join

Translate

PSA: EPYC 7002 CPUs may hang after 1042 days of uptime

The April 2023 Epyc 2nd gen revision guide has errata #1474:

Description

A core will fail to exit CC6 after about 1044 days after the last system reset. The time of failure may vary depending on the spread spectrum and REFCLK frequency.

Potential Effect on System

A core will hang.

Suggested Workaround

Either disable CC6 or reboot system before the projected time of failure.

Fix Planned

No fix planned

Despite what they say, the problem actually manifests at 1042 days and roughly 12 hours. The TSC ticks at 2800 MHz, and 2800 * 10**6 * 1042.5 days almost equals 0x380000000000000, which has too many zeros not to be a coincidence.

Note that your server will almost definitely hang, requiring a physical (or IPMI) reboot, because no interrupts, including NMIs, can be delivered to the zombie cores: this means no scheduler, no IPIs, nothing will work.

Read more

259 80 Share

Pentium FDIV bug

From Wikipedia, the free encyclopedia

The **Pentium FDIV bug** is a hardware bug affecting the floating-point unit (FPU) of the early Intel Pentium processors. Because of the bug, the processor would return incorrect binary floating point results when dividing certain pairs of high-precision numbers. The bug was discovered in 1994 by Thomas R. Nicely, a professor of mathematics at Lynchburg College.[1] Missing values in a lookup table used by the FPU's floating-point division algorithm led to calculations acquiring small errors. While these errors would in most use-cases only occur rarely and result in small deviations from the correct output values, in certain circumstances the errors can occur frequently and lead to more significant deviations.[2]

66 MHz Intel Pentium (sSpec=SX837) with the FDIV bug

The severity of the FDIV bug is debated. Though rarely encountered by most users (*Byte* magazine estimated that 1 in 9 billion floating point divides with random parameters would produce inaccurate results),[3] both the flaw and Intel's initial handling of the matter were heavily criticized by the tech community.

LWN.net
News from the source

Content
Weekly Edition
Archives
Search
Kernel
Security
Events calendar
Unread comments

LWN FAQ
Write for us

Edition
Return to the Briefs page

User: Password: Log in | Subscribe | Register

Intel Skylake/Kaby Lake processors: broken hyper-threading

Henrique de Moraes Holschuh has posted an advisory about a processor/microcode defect recently identified on Intel Skylake and Intel Kaby Lake processors with hyper-threading enabled. "TL;DR: unfixed Skylake and Kaby Lake processors could, in some situations, dangerously misbehave when hyper-threading is enabled. Disable hyper-threading immediately in BIOS/UEFI to work around the problem. Read this advisory for instructions about an Intel-provided fix."

From:	Henrique de Moraes Holschuh <hmh-AT-debian.org>
To:	debian-user-AT-lists.debian.org, debian-devel-AT-lists.debian.org
Subject:	[WARNING] Intel Skylake/Kaby Lake processors: broken hyper-threading
Date:	Sun, 25 Jun 2017 09:19:36 -0300
Message-ID:	<20170625121936.GA7714@khazad-dum.debian.net>

EXTREMETECH Search Extremetech SEARCH

Computing Phones Security Gaming Science Space Deep Dives Deals Shop

HOME › EXTREME › AMD PHENOM, BARCELONA CHIPS HIT BY LOCK-UP BUG

AMD Phenom, Barcelona Chips Hit By Lock-up Bug

By Mark Hachman on December 5, 2007 at 4:54 pm | Comments

This site may earn affiliate commissions from the links on this page. Terms of use.

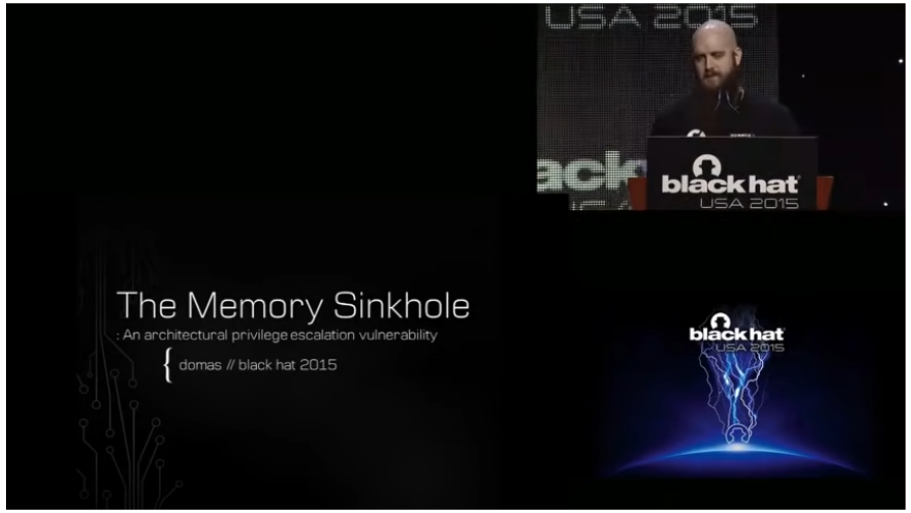
AMD has confirmed a bug that can cause its new Phenom and existing Barcelona processors to lock up.

Specifically, AMD has found a bug in the translation-lookaside buffer, which can impact some of AMD's quad-core chips. AMD will rework both chips and provide a new stepping, sources said, but in the meantime motherboard manufacturers are being asked to distribute a BIOS patch that, unfortunately, reportedly cuts performance by about 10 percent.

"You may remember that during our Q3 earnings call, AMD acknowledged that our initial ramp of Barcelona had been slower than anticipated," AMD spokesman Phil Hughes said in an emailed statement. "However we did say during that call that we would ship 'hundreds of thousands of quad-core processors' into the server and desktop segments during Q4. AMD is tracking to this guidance. Quad Core AMD Opteron processor is the most advanced x86 processor ever introduced to the market and as such there are design and process tuning steps that will take longer than expected.

"There has been some talk about an erratum relative to our TLB cache in Barcelona as well as Phenom processors resulting in delays," Hughes added. "AMD notified customers of this erratum and released a BIOS fix prior to the Nov. 19th launch that resolves it. We are experiencing strong AMD Phenom demand and are shipping parts to channel, system builders and OEM customers."

YouTube HK 搜索

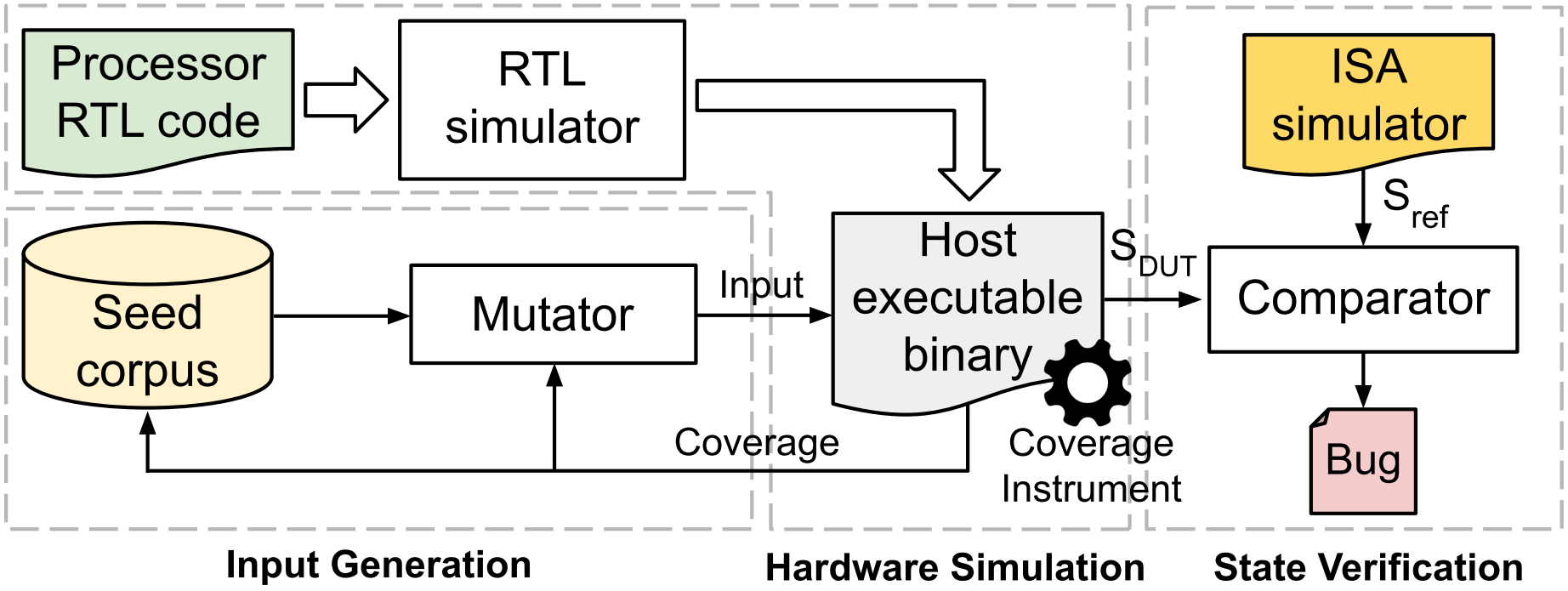


The Memory Sinkhole - Unleashing An X86 Design Flaw Allowing Universal Privilege Escalation

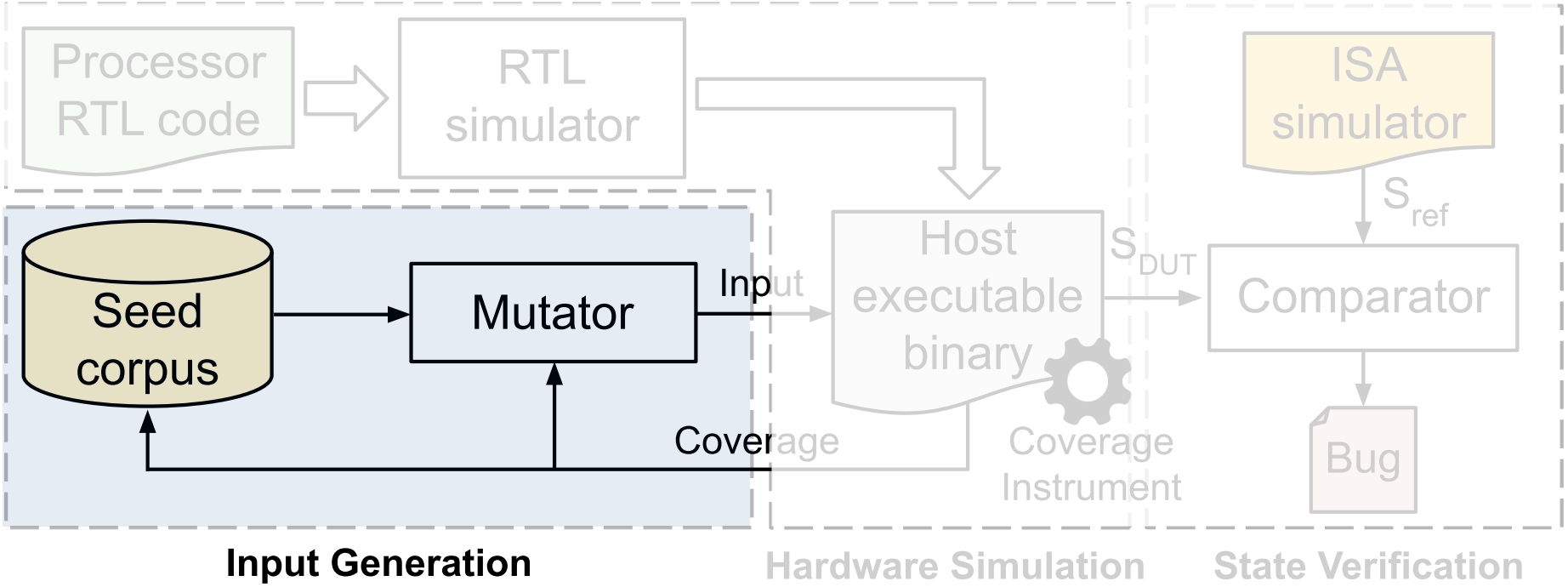
Black Hat
19.9万位订阅者
订阅

4078 分享 下载

Processor Fuzzing



Processor Fuzzing




```
"10111111
000001010
011100111
01110..."
```

seed

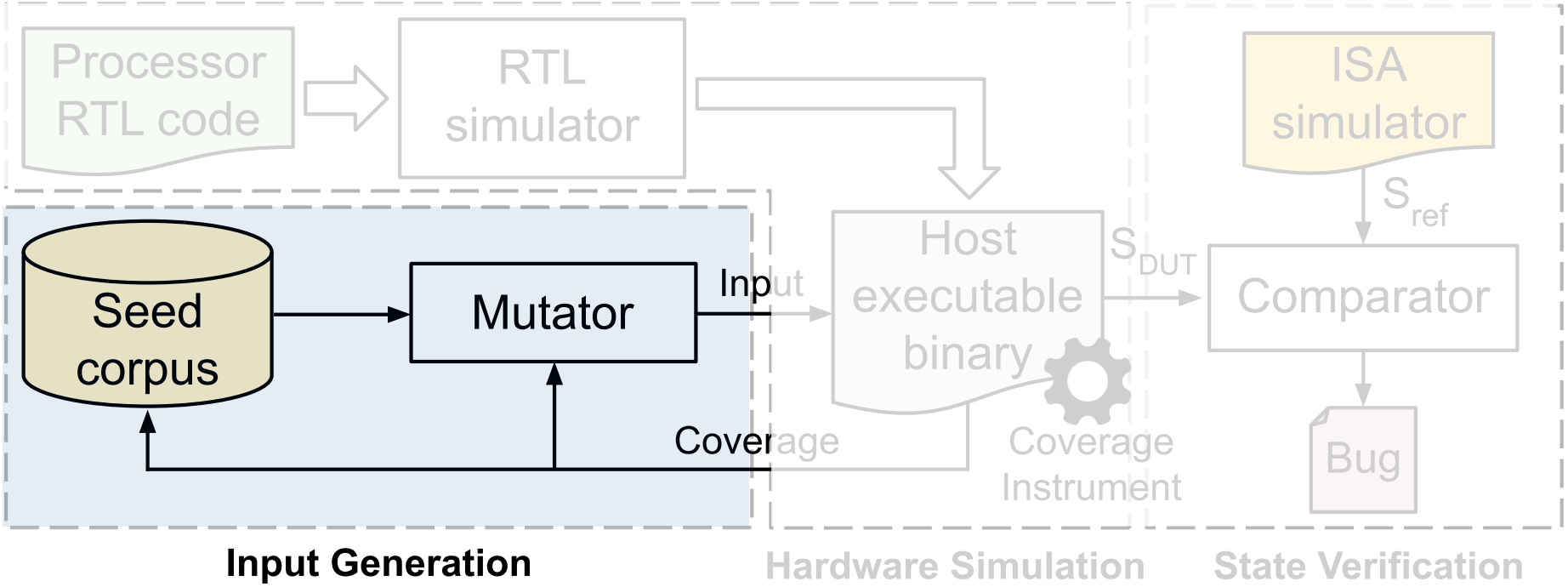
```
slli    a2,a2,0x4
li      a0,0
lui     a2,0xef1
...     ...     ...
```

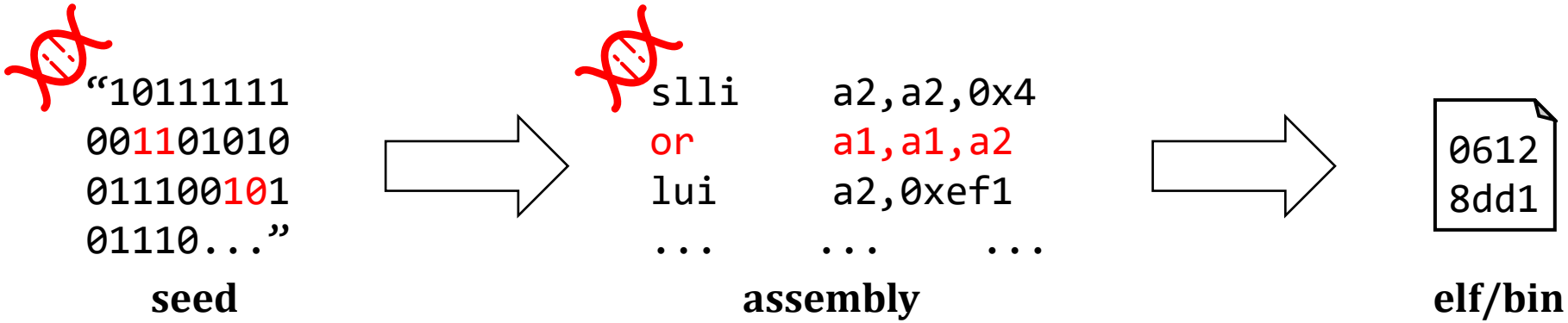
assembly

```
0612
4501
```

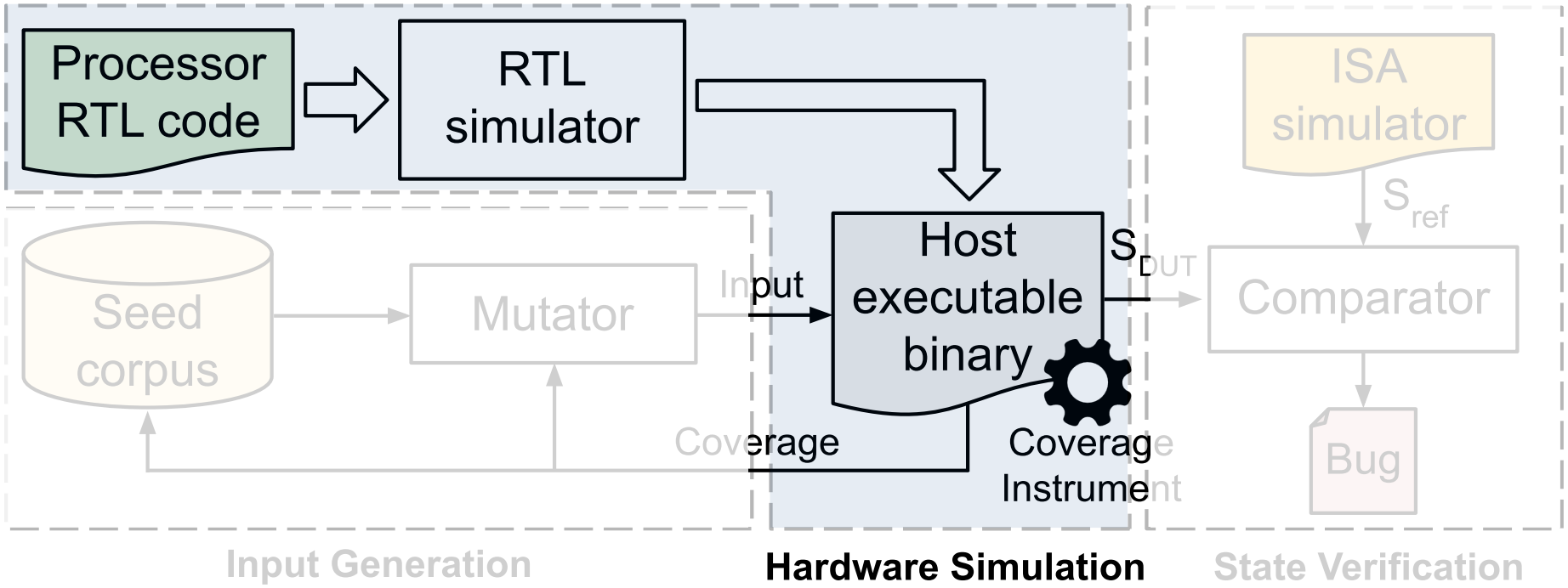
elf/bin

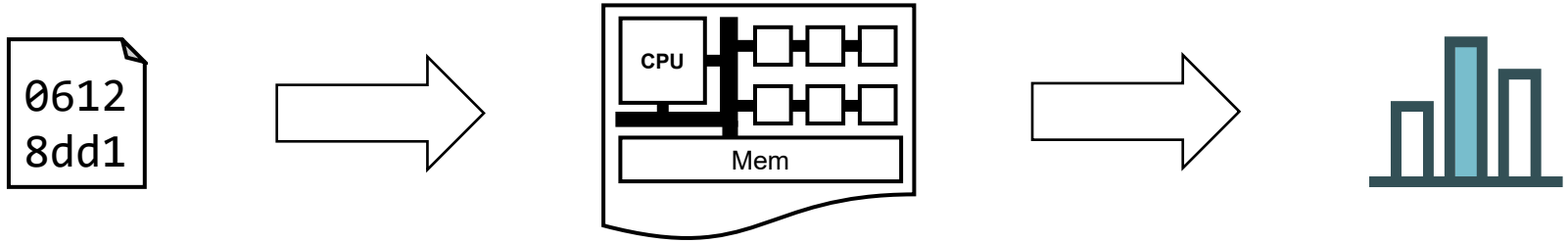
Processor Fuzzing





Processor Fuzzing



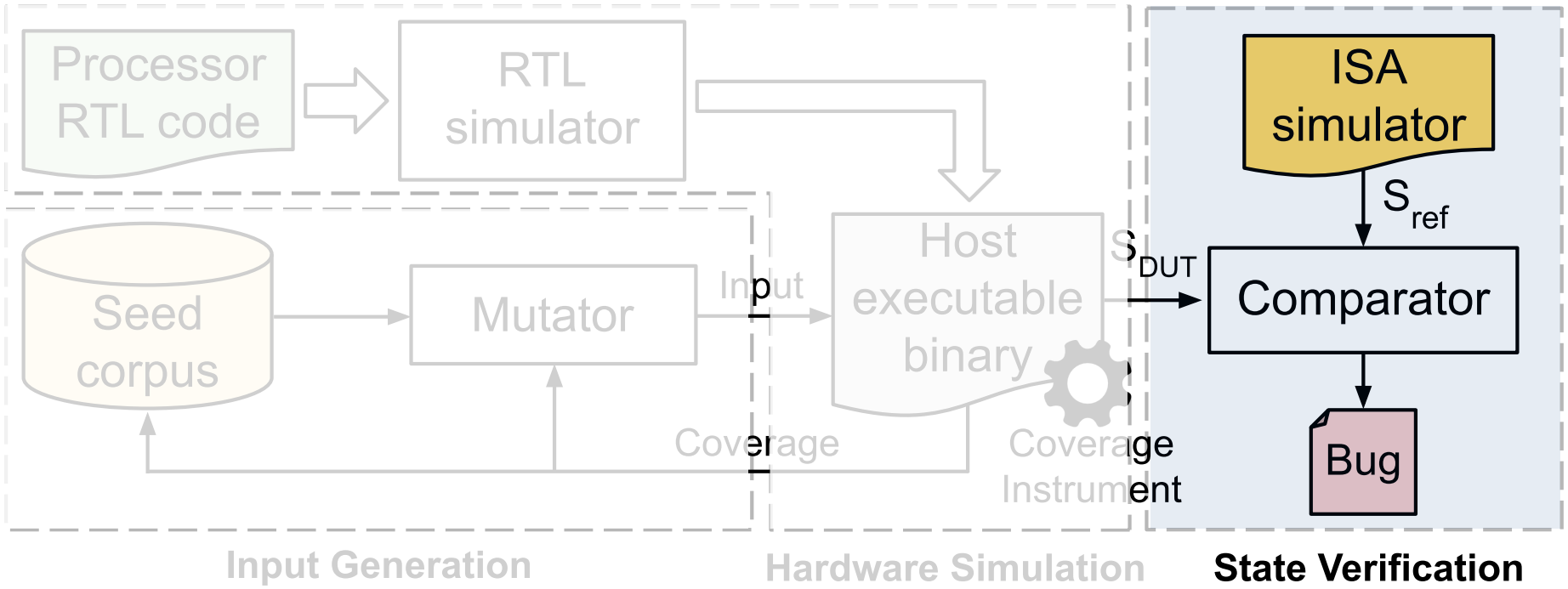


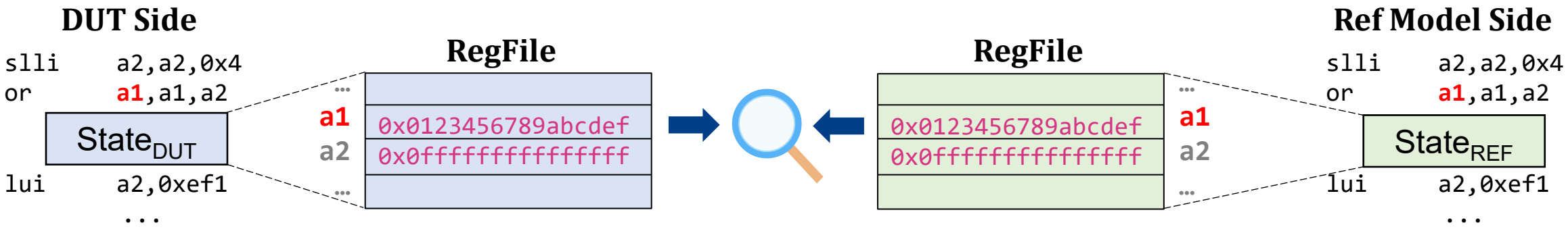
elf/bin

DUT Software Model

Coverage

Processor Fuzzing

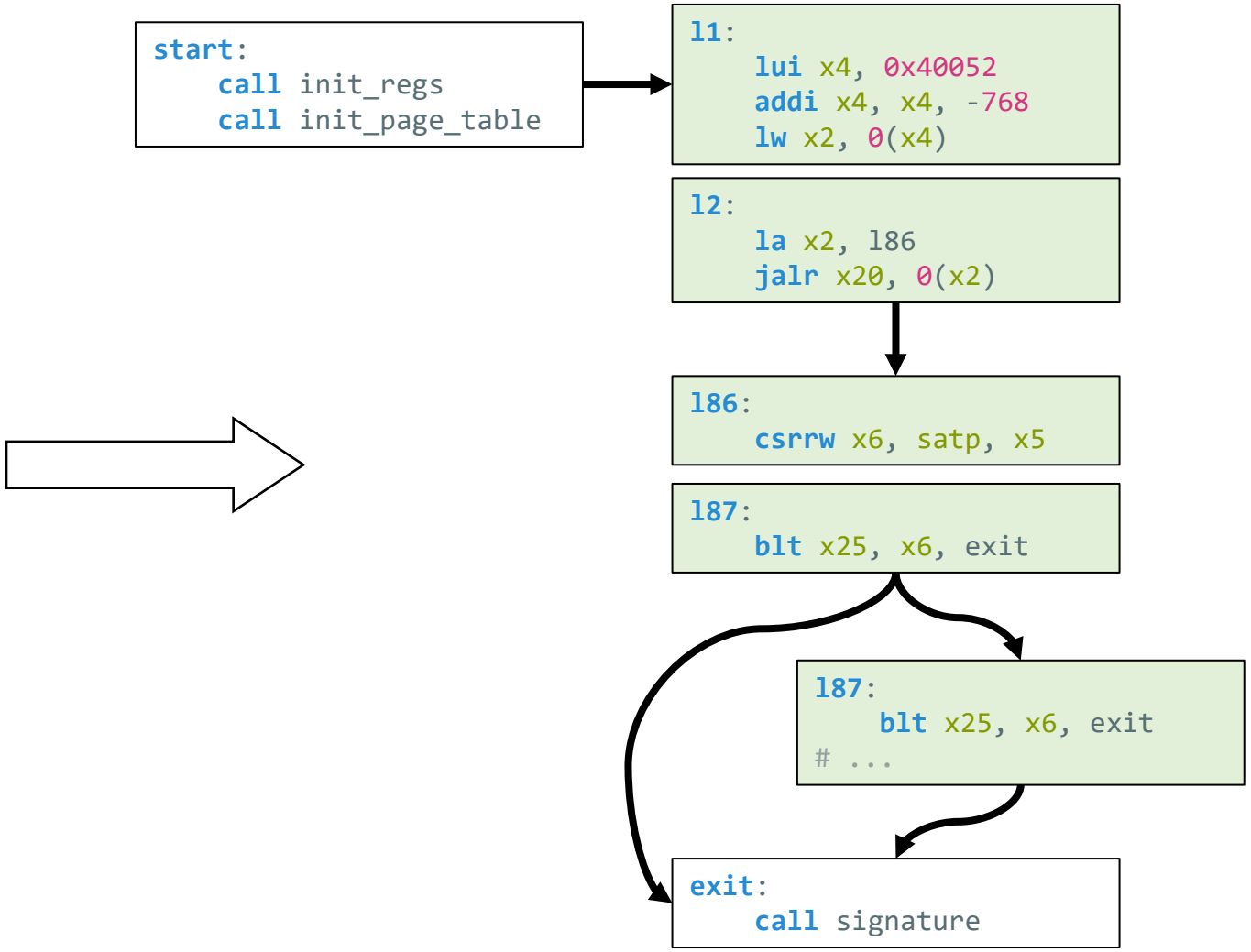




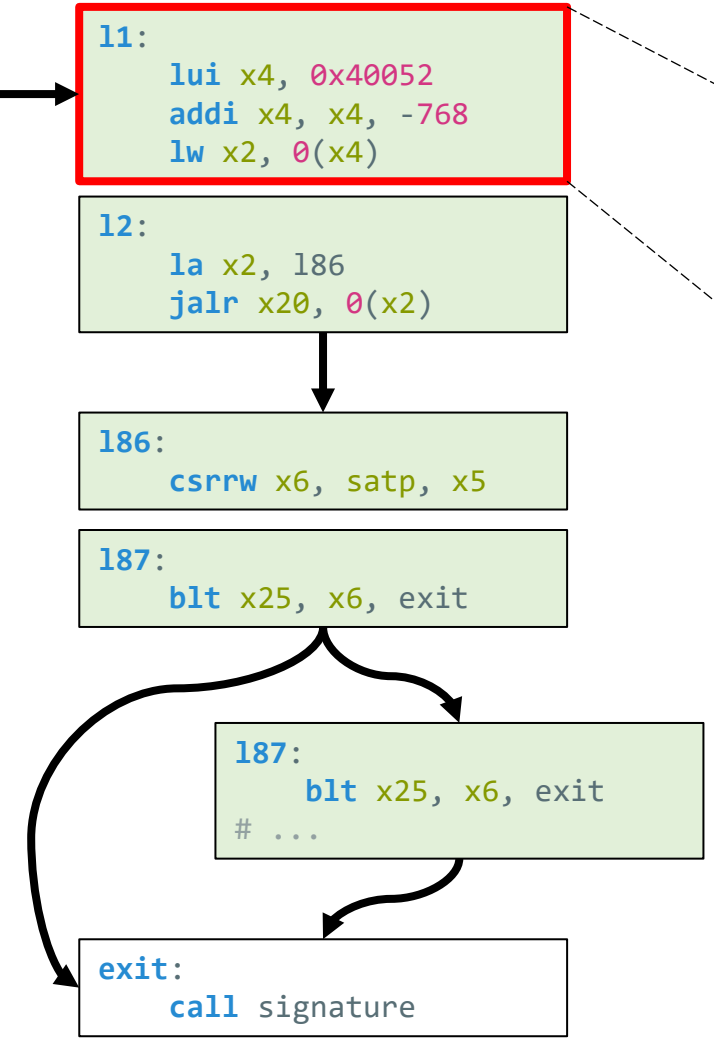
Challenges of Processor Fuzzing

DifuzzRTL:

```
1  start:
2      call init_regs
3      call init_page_table
4  l1:
5      lui x4, 0x40052
6      addi x4, x4, -768
7      lw x2, 0(x4)
8  l2:
9      la x2, l86
10     jalr x20, 0(x2)
11     # ...
12 l86:
13     csrrw x6, satp, x5
14 l87:
15     blt x25, x6, exit
16     # ...
17 exit:
18     call signature
```



Challenges of Processor Fuzzing



1. Complex Input Grammar

```
l1:
    lui x4, 0x40052
    addi x4, x4, -768
    lw x2, 0(x4)
```

- Read the base address from x4
- Calculate the effective address by adding x4 to the offset
- Load different length of value from the effective address
- Save the value from memory to x2

C1: Processor State

x4 ∈ {memory range}

C2: Instruction Field

lb, lh, lw, ld,
lbu, lhu, lwu, **Reserved**

C3: Program Semantic

(x4 + offset) % 4 == 0

Challenges of Processor Fuzzing

2. Deceptive Mutation Guidance

```
l1:
    lui x4, 0x40052
    addi x4, x4, -768
    lw x2, 0(x4)
```

```
l2:
    la x2, l86
    jalr x20, 0(x2)
```

```
l86:
    csrrw x6, satp, x5
```

```
l87:
    blt x25, x6, exit
```

```
l87:
    blt x25, x6, exit
# ...
```

```
exit:
    call signature
```

```
l2:
    la x2, l86
    jalr x20, 0(x2)
```

```
l3:
    xori x23, x22, -2047
l4:
    fsgnj.s f19, f4, f5
# ...
```

```
l86:
    csrrw x6, satp, x5
```

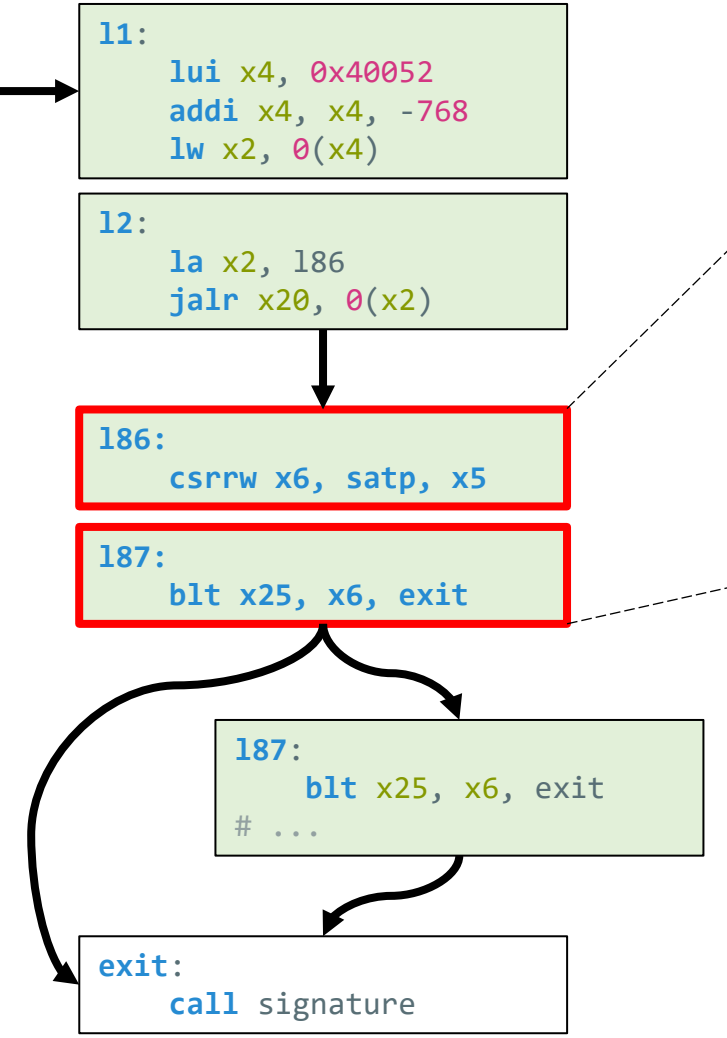
Generated Instructions

$\neq$

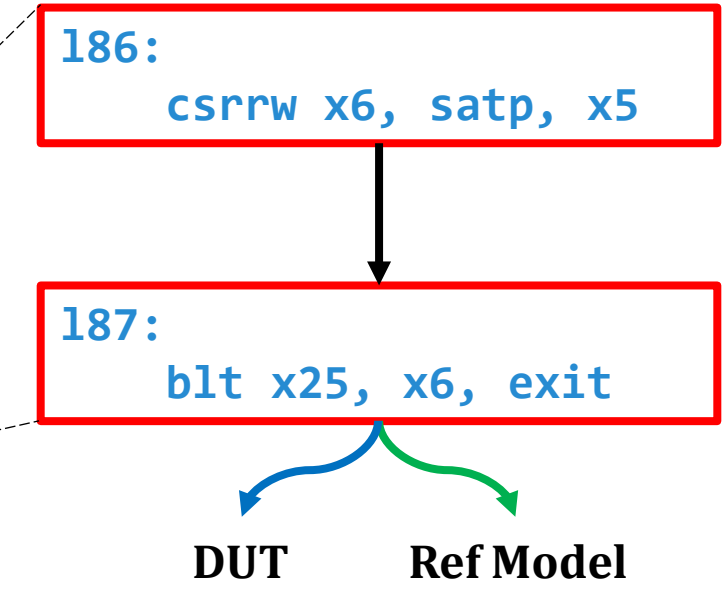
Executed Instructions

Unexecuted valuable mutations will be discarded

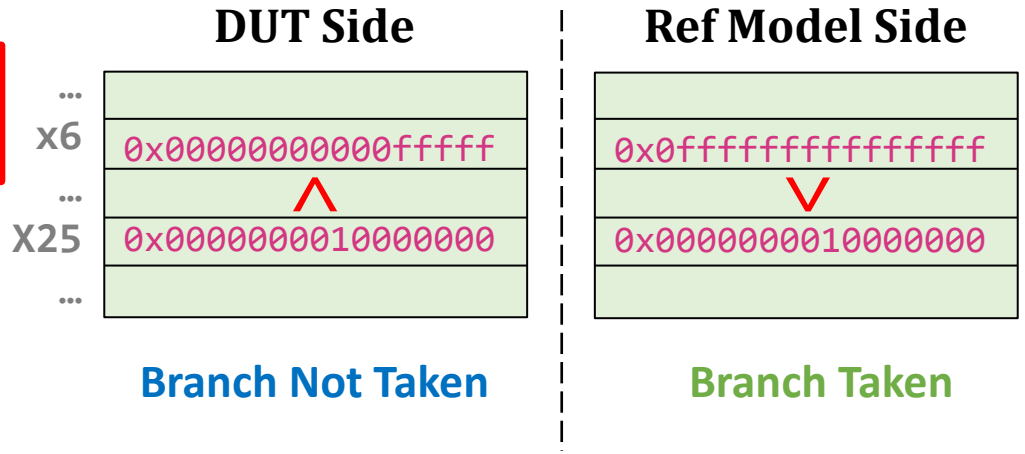
Challenges of Processor Fuzzing



3. Model Implementation Differences



CSR satp has **WARL**(Write Any Values, Reads Legal Values) fields

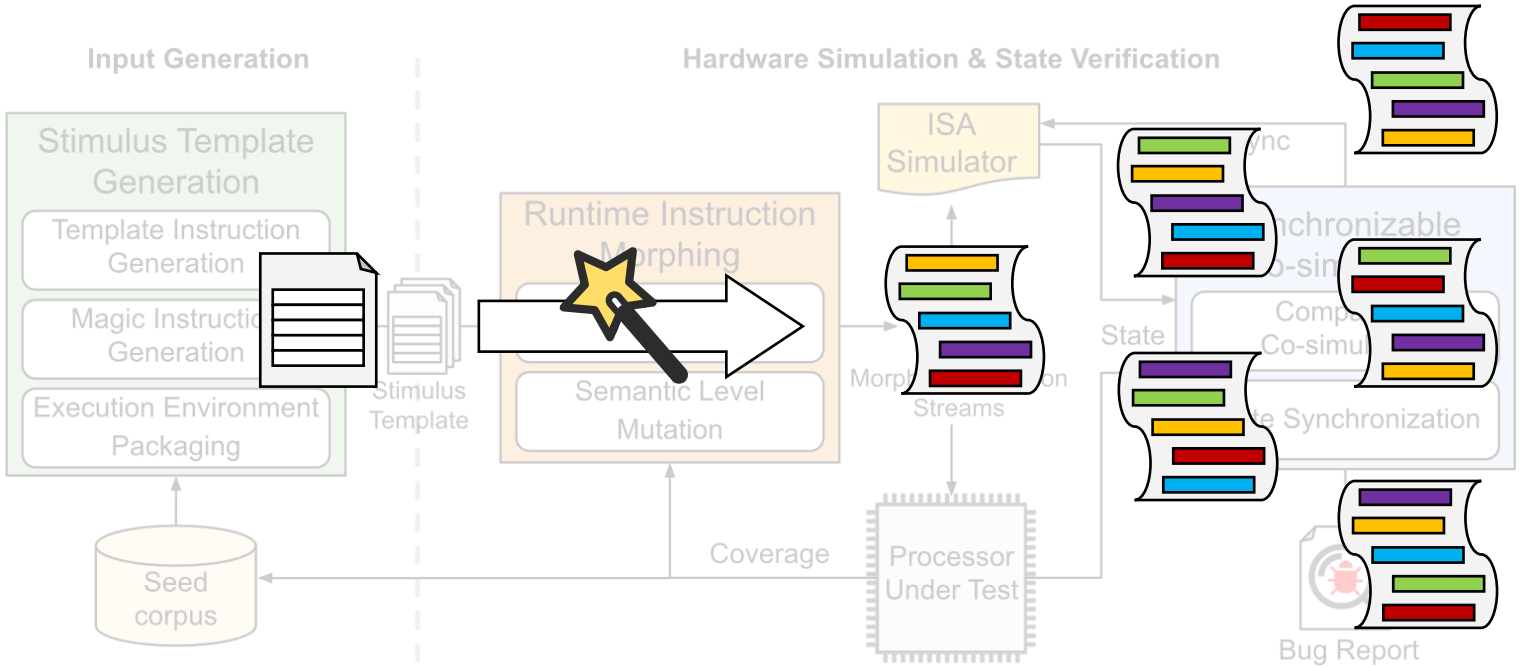


Divergent control flows make subsequent execution meaningless

Insight

Mutating those instructions that are going to be executed

- all mutations are executed, yielding effective coverage
- use runtime context to simplify input generation



Stimulus Template Generation

DifuzzRTL:

```
l1:
    lui x4, 0x40052
    addi x4, x4, -768
    lw x2, 0(x4)
```

MorFuzz:

```
fuzztext_ls_27:
# magic inst
    ld x1, RDM_DATA_ADDR(x0)
# template inst
    lh x??, ??(x??)
```

Magic Instruction

- load a random value with desired type into target register
- processor state mutation primitive

Template Instruction

- blank instruction with dummy fields
- instruction field & program semantic mutation primitive

Runtime Instruction Morphing

imm	rs1	funct3	rd	opcode
????????????	?????	001	?????	0000011

```
lh x??, ??(x??)
```

Field Level Mutation

- identify instruction format
- mutate instruction opcode field

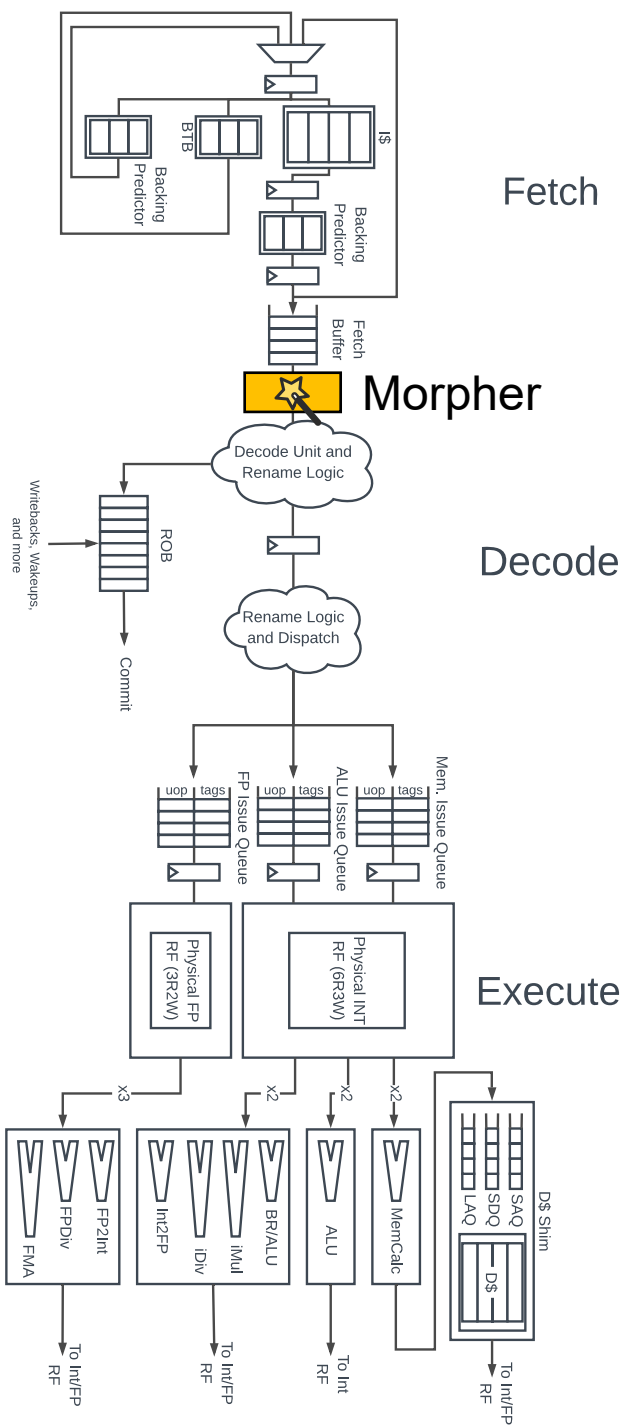
????????????	?????	010	?????	0000011

Semantic Level Mutation

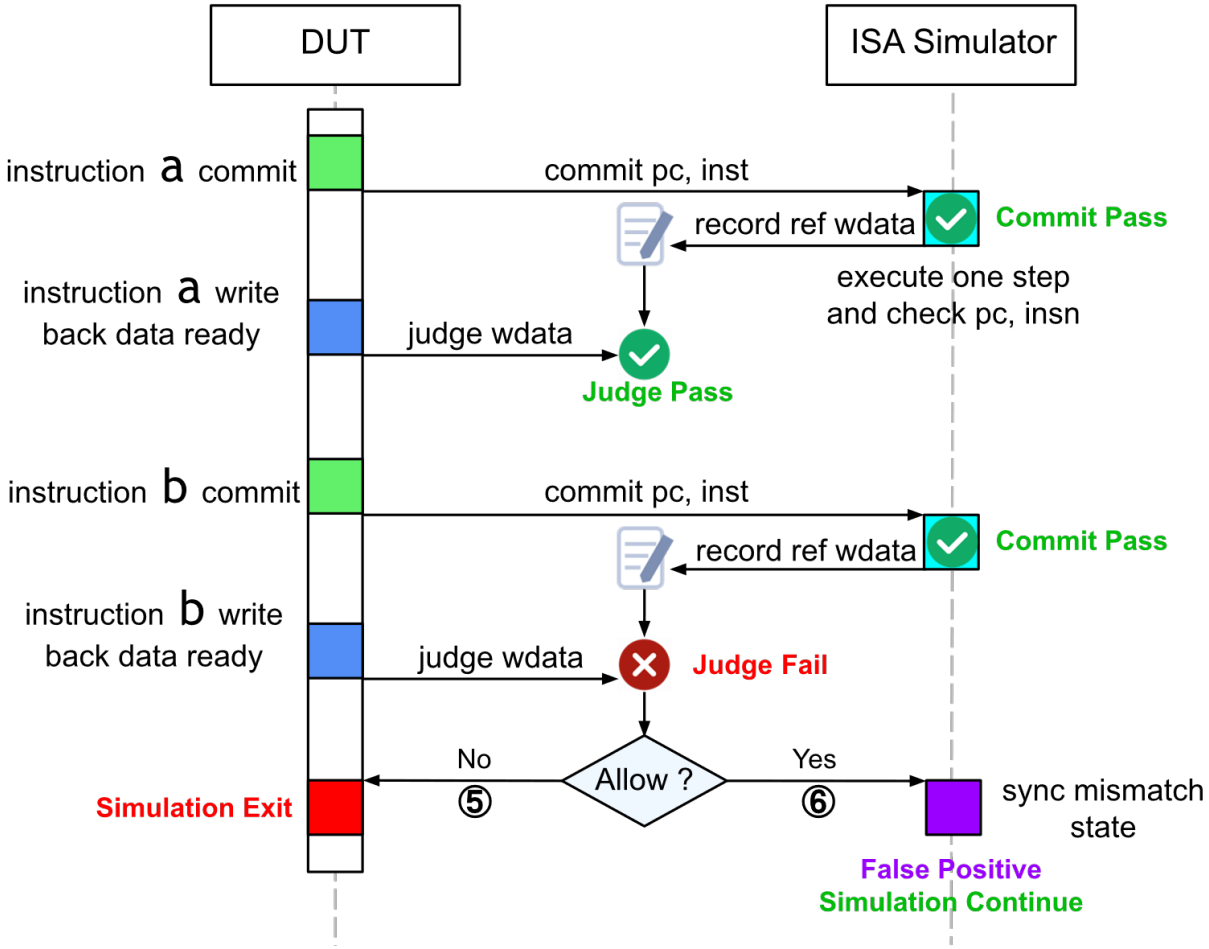
- select register with desired type
- calculate the other fields

000000000100	00001	010	00010	0000011

```
lw x2, 4(x1)
```



Synchronizable Co-simulation



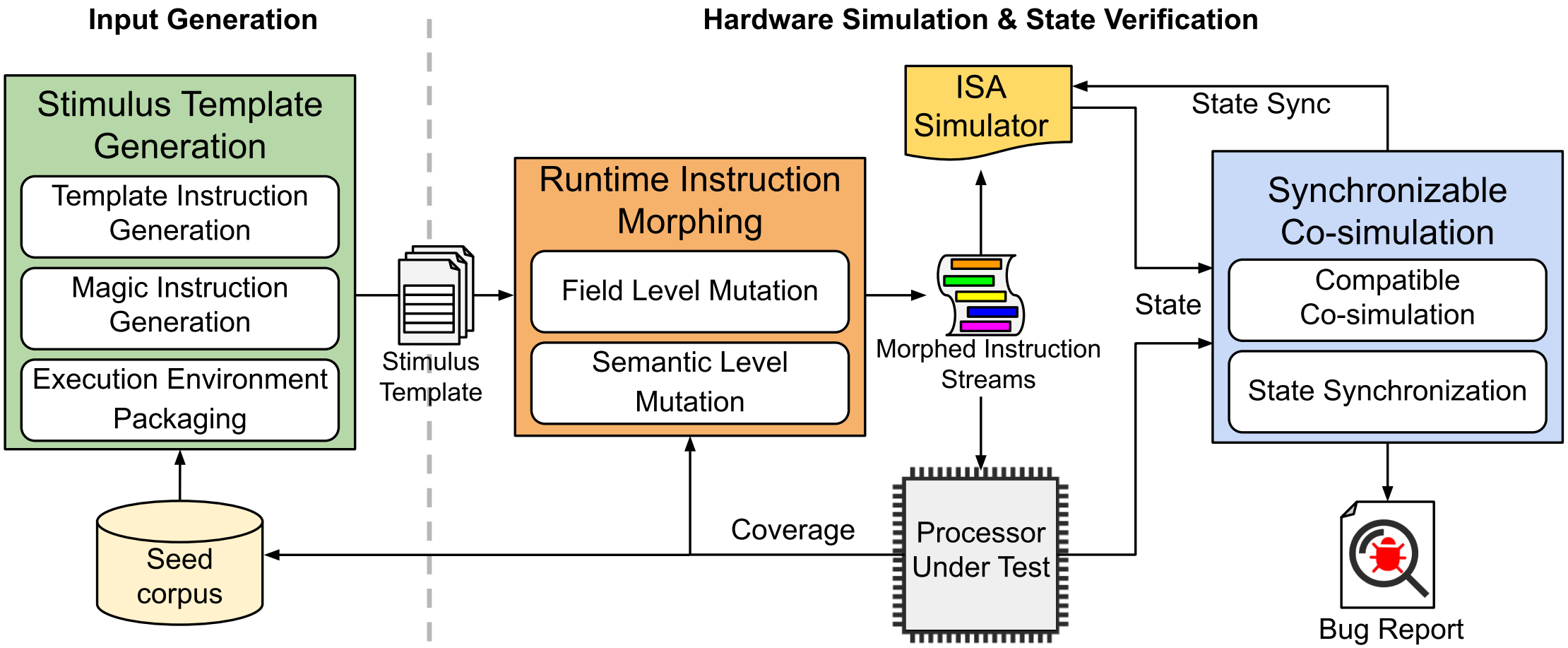
Commit Stage

- DUT commits control flow info
- REF execute one step
- **cross-check control flow info**
  - **match**, continue
  - **mismatch**, report as bug
- REF commits reference write-back data

Judge Stage

- DUT finally commits write-back data
- **cross-check wdata**
  - **match**, continue
  - **mismatch**, analysis committed info
    - **sync DUT state to REF if permitted**
    - otherwise report as bug

MorFuzz


Input Generation
Hardware Simulation & State Verification
Stimulus Template Generation
Template Instruction Generation
Magic Instruction Generation
Execution Environment Packaging
Stimulus Template
Runtime Instruction Morphing
Field Level Mutation
Semantic Level Mutation
ISA Simulator
State Sync
Morphed Instruction Streams
State
Synchronizable Co-simulation
Compatible Co-simulation
State Synchronization
Seed corpus
Coverage
Processor Under Test
Bug Report

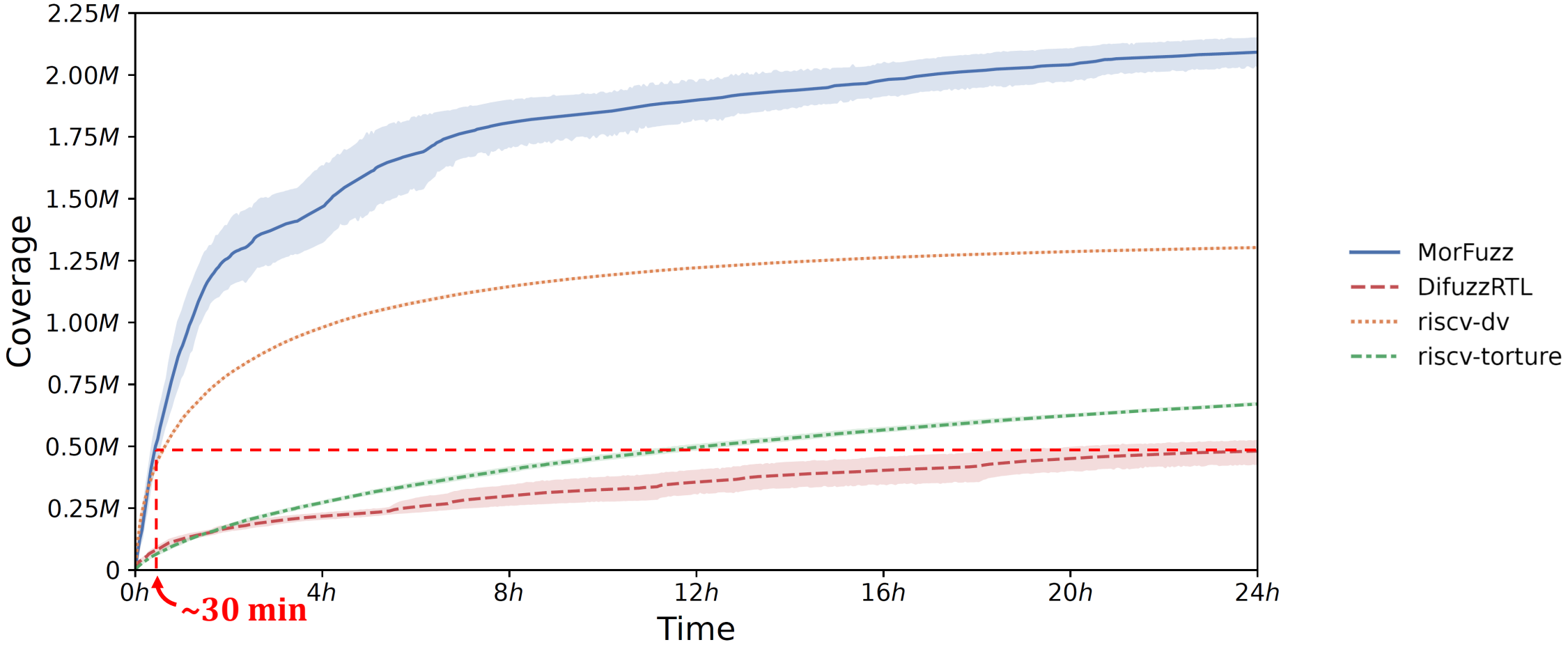
Bug Detected

MorFuzz detected 19 bugs including 17 new bugs, 13 CVEs

Processor	Bug Description	CVE/Issue ID	CWE	New Bug	Confirmed	Fixed
Rocket	**B1**: Treat `aes64ksli` with rcon greater than `0xA` as valid	CVE-2022-34632	CWE-327	✓	✓	✓
	B2: Error in condition of the `rocc_illegal` signal	Issue #2980	CWE-1281	✓	✓	✓
	B3: The `vsstatus.xs` is writable	CVE-2022-34627	CWE-732	✓	✓	✓
BOOM	**B4**: Incorrect exception type when a PMA violation	CVE-2022-34636	CWE-1202	✓	✓	
	B5: Incorrect exception type when a PMP violation	CVE-2022-34641	CWE-1198	✓	✓	
	B6: Floating-point instruction with invalid `rm` field does not raise exception	Issue #458	CWE-391		✓	
	B7: Floating-point instruction with invalid `frm` does not raise exception	Issue #492	CWE-391		✓	
CVA6	**B8**: Crafted or incorrectly formatted `sfence.vma` instructions are executed	CVE-2022-34633	CWE-1242	✓	✓	✓
	B9: Crafted or incorrectly formatted `dret` instructions are executed	CVE-2022-34634	CWE-1242	✓	✓	✓
	B10: Non-standard `fence` instructions are treated as illegal	CVE-2022-34639	CWE-1209	✓	✓	✓
	B11: The `mstatus.sd` field does not update immediately	CVE-2022-34635	CWE-1199	✓	✓	
	B12: The value of `mtval`/`stval` after `ecall`/`ebreak` is incorrect	CVE-2022-34640	CWE-755	✓	✓	
	B13: Incorrect exception type when a PMA violation	CVE-2022-34636	CWE-1202	✓	✓	
	B14: Incorrect exception type when a PMP violation	CVE-2022-34641	CWE-1198	✓	✓	✓
	B15: Incorrect exception type when accessing an illegal virtual address	CVE-2022-34637	CWE-754	✓	✓	
	B16: Improper physical PC truncate	Issue #901	CWE-222	✓	✓	
	B17: Incorrect `lr` exception type	CVE-2022-37182	CWE-754	✓	✓	
Spike	**B18**: The component `mcontrol.action` contains the incorrect mask	CVE-2022-34642	CWE-787	✓	✓	✓
	B19: Incorrect exception priotrity when accessing memory	CVE-2022-34643	CWE-754	✓	✓	✓

Coverage

4.4× than DifuzzRTL, 3.1× than riscv-torture, 1.6× than riscv-dv



Conclusion

MorFuzz is a novel Processor Fuzzer

- detect architecture functional bugs automatically
  - instruction morphing effectively guides fuzzing
  - state synchronization eliminates false positive
- thorough evaluation on real world processors
  - faster & higher coverage than SOTA
  - detected 19 bugs with 13 CVEs assigned
- battle-hardened and open-source
  - deployed in a undergraduate CPU design course (50+ students)
  - 1st Place in HACK@DAC 2023
  - https://github.com/sycuricon/MorFuzz

Thank You!

Jinyan Xu

phantom@zju.edu.cn