Fuzzing for Discovering Bugs and Side Channels in Processors

Chathura Rajapaksha[1], Sadullah Canakci[1], Leila Delshadtehrani[1], Anoop Nataraja[2], Michael Bedford Taylor[2], Manuel Egele[1], Ajay Joshi[1]

[1]Department of ECE, Boston University

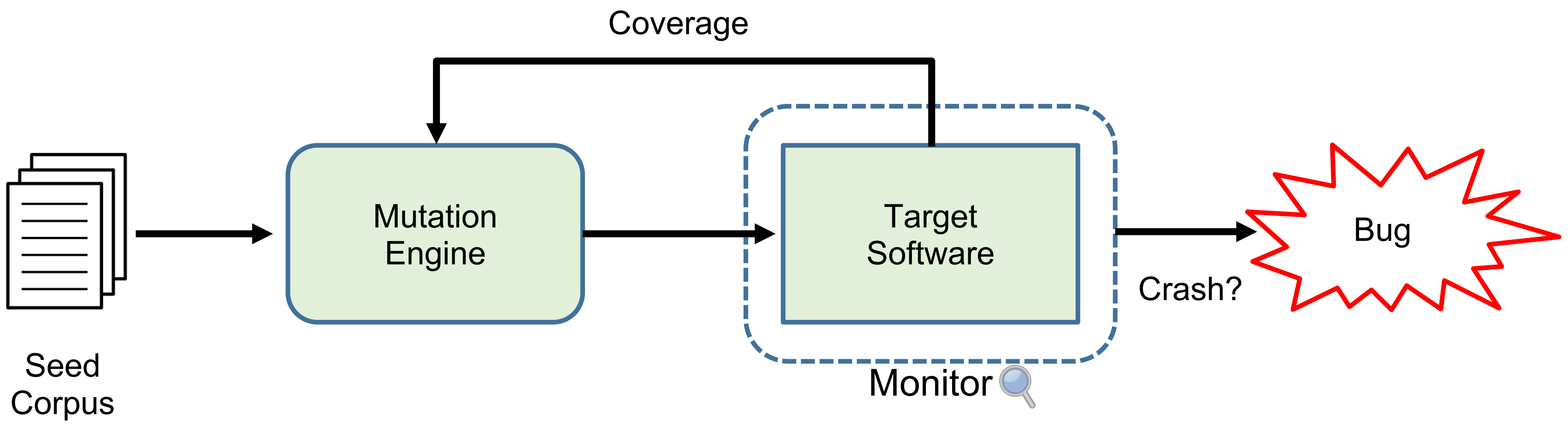
[2]Department of ECE, University of Washington

Outline

- What is Fuzzing?
- ProcessorFuzz: Processor Fuzzing with Control and Status Registers Guidance [1]
- SIGFuzz: A Framework for Discovering Microarchitectural Timing Side Channels [2]
- Summary

What is Fuzzing?

- Fuzzing/fuzz testing:
  - Running the target software with random or mutated inputs.

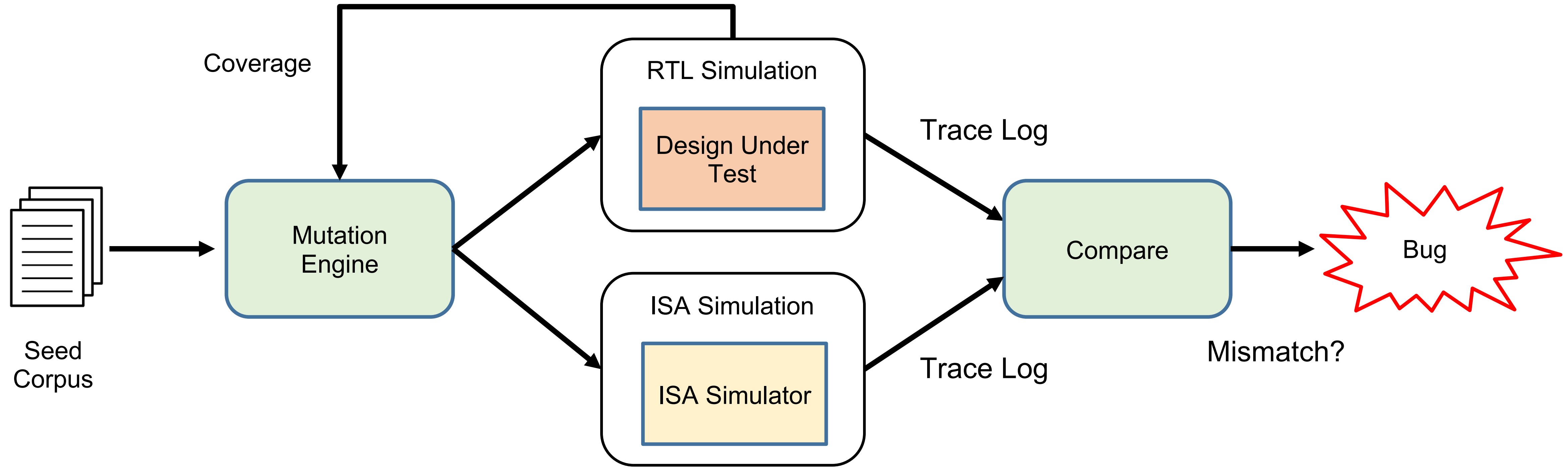


Adapting Fuzzing for Hardware Testing

- What is the input format and how to mutate the inputs?
  - Driving RTL signals Vs assembly test programs
  - Mutations
- What is the coverage feedback metric?
  - Standard RTL coverage metrics Vs new coverage metrics for fuzzing
- How to detect when the **bugs** get triggered?
  - Golden models
  - Hardware equivalent of a software crash?

Fuzzing for Hardware Testing

- Fuzzing has been ‘recently’ adapted for hardware testing.
- Hardware fuzzing research is rapidly growing [5-7].

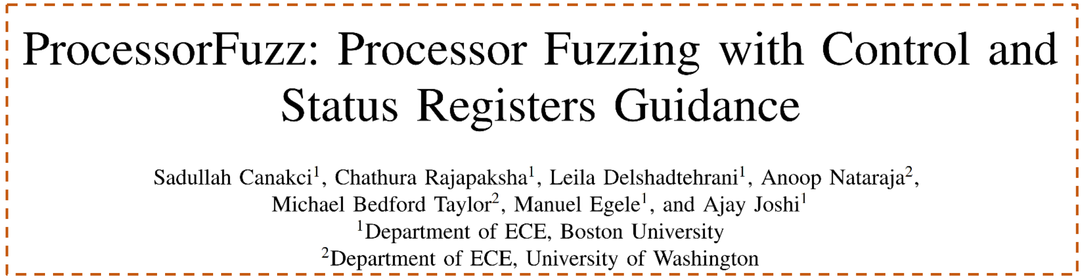


Our Contributions

- What is the input format and how to mutate it?
  - Driving RTL signals Vs assembly test programs
  - Mutations

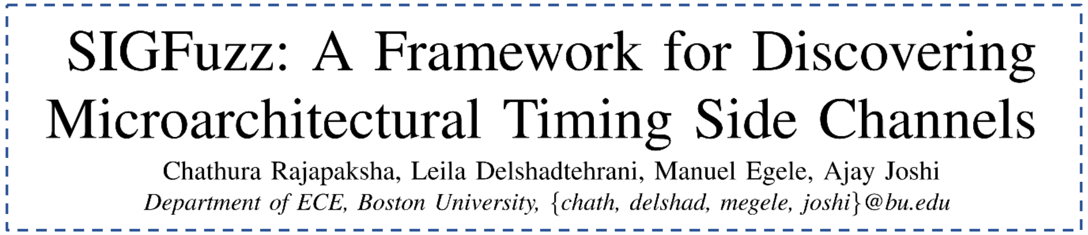
- What is the coverage feedback metric?
  - Standard RTL coverage metrics Vs. New coverage metrics for fuzzing

- How to detect when the **bugs** get triggered?
  - Golden models
  - Hardware equivalent of a software crash?



ProcessorFuzz: Processor Fuzzing with Control and Status Registers Guidance

Sadullah Canakci[1], Chathura Rajapaksha[1], Leila Delshadtehrani[1], Anoop Nataraja[2], Michael Bedford Taylor[2], Manuel Egele[1], and Ajay Joshi[1]
[1]Department of ECE, Boston University
[2]Department of ECE, University of Washington

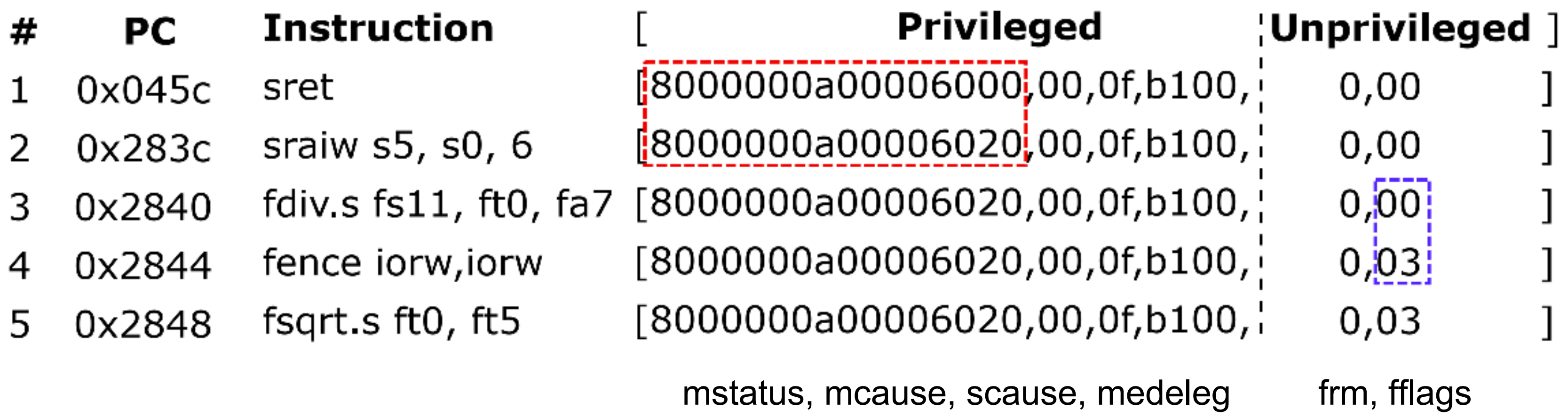


SIGFuzz: A Framework for Discovering Microarchitectural Timing Side Channels

Chathura Rajapaksha, Leila Delshadtehrani, Manuel Egele, Ajay Joshi
Department of ECE, Boston University, {chath, delshad, megele, joshi}@bu.edu

ProcessorFuzz: Processor Fuzzing with Control and Status Registers Guidance

- Problem: Existing coverage metrics for processor fuzzing is limited by
  - Lack of support for widely used Hardware Description Languages (HDLs)
  - Misleading coverage feedback
- Introduces a new coverage metric for processor fuzzing.
  - New transition in CSR values -> coverage increase



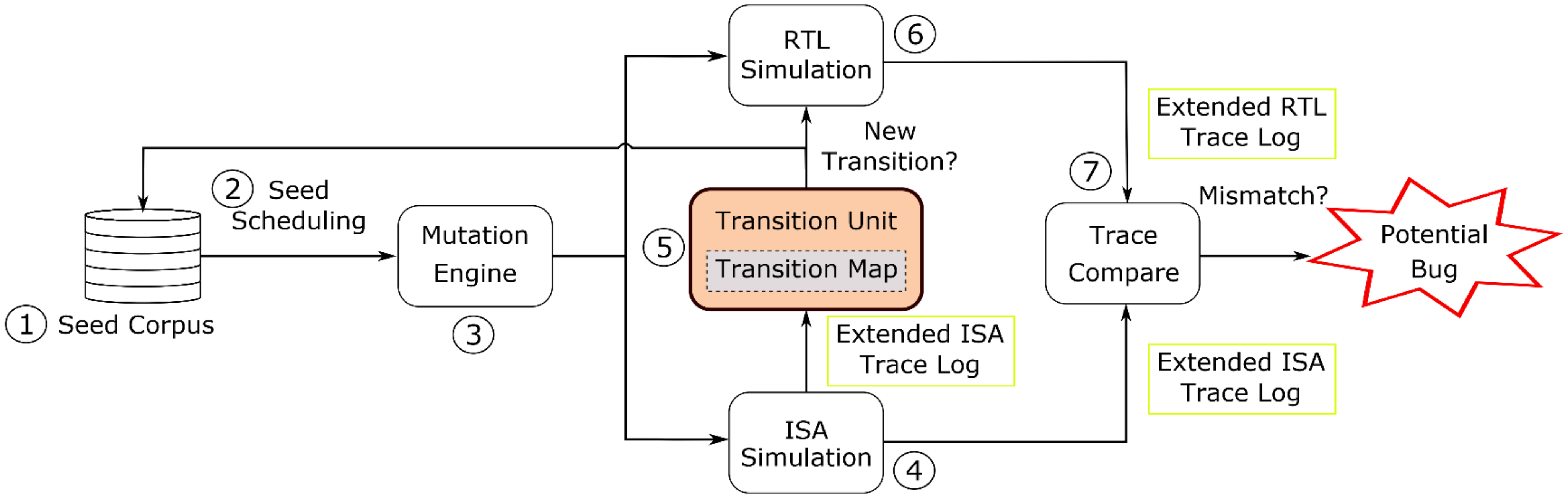
#	PC	Instruction	[Privileged	Unprivileged]
1	0x045c	sret	[8000000a00006000,00,0f,b100,	0,00]
2	0x283c	sraiw s5, s0, 6	[8000000a00006020,00,0f,b100,	0,00]
3	0x2840	fdiv.s fs11, ft0, fa7	[8000000a00006020,00,0f,b100,	0,00]
4	0x2844	fence iorw,iorw	[8000000a00006020,00,0f,b100,	0,03]
5	0x2848	fsqrt.s ft0, ft5	[8000000a00006020,00,0f,b100,	0,03]

mstatus, mcause, scause, medeleg

frm, fflags

ProcessorFuzz: Design Overview

- ProcessorFuzz uses an ISA simulator to collect CSR transition coverage, making the coverage collection more efficient and HDL agnostic.



ProcessorFuzz: Evaluation

- Evaluated on Rocket [8], BOOM [9], and BlackParrot [10] RISC-V processors.
- Detect known bugs 23% faster than the state-of-the-art DifuzzRTL [7].
- Discovered 9 new bugs
  - 6 in BlackParrot processor
  - 2 in Rocket and BOOM processors
  - 1 in Dromajo ISA simulator

SIGFuzz: A Framework for Discovering Microarchitectural Timing Side Channels

- Problem: Existing methods are limited in
  - Scalability
  - Scope of side channels they can discover
- SIGFuzz introduces a generic method for discovering microarchitectural timing side channels.

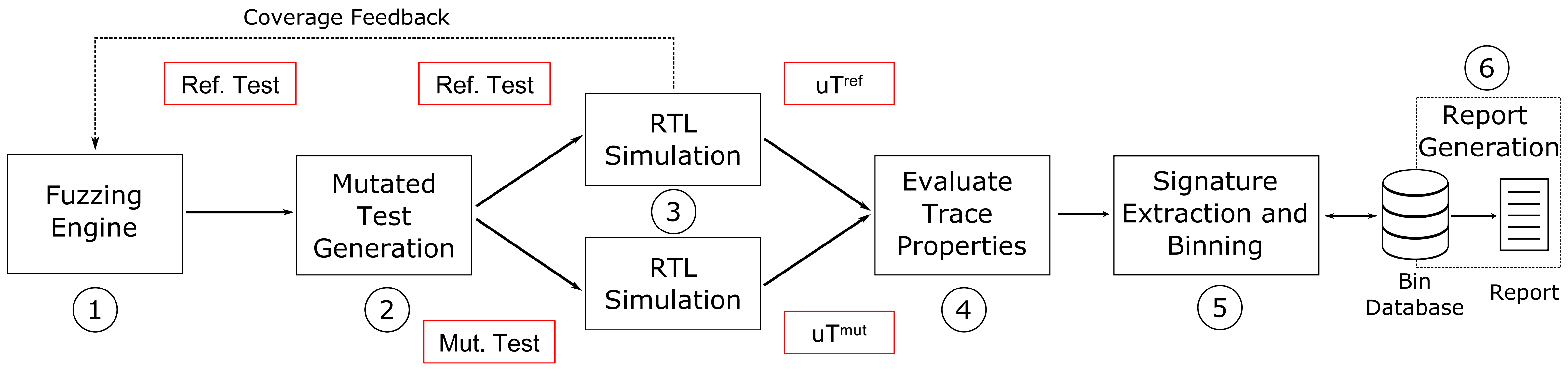
	Reference Test	Mutated Test	
t^{ref}_1	1: …	1: …	t^{mut}_1
t^{ref}_2	2: div x1, x5, x6	2: div x1, x5, x6	t^{mut}_2
t^{ref}_3	3: **mul x8, x15, x21**	3: ***nop***	t^{mut}_3
t^{ref}_4	4: …	4: …	t^{mut}_4

$t^{ref}_2 \neq t^{mut}_2 \Rightarrow$ Potential Side Channel

Trace Property 1

SIGFuzz: Design Overview

- SIGFuzz flags potential timing side channels using trace properties evaluated on cycle-accurate commit traces.



SIGFuzz: Evaluation

- Evaluated on Rocket and BOOM RISC-V processors.
- Discovered both known and new timing side channels.
  - 3 new side channels
  - 2 known side channels
- Spectre-style attack based on a newly discovered side channel on BOOM

Summary

- Fuzzing is a proven software testing technique that is recently adapted for hardware testing.

- **ProcessorFuzz** introduces a new coverage metric based on CSRs, which improves the overall efficiency of processor fuzzing.
  - ProcessorFuzz discovered 8 new bugs in Rocket, BOOM, and BlackParrot processors.
  - ProcessorFuzz will be open-sourced soon: https://github.com/bu-icsg/ProcessorFuzz

- **SIGFuzz** introduces a generic method for discovering a broader scope of microarchitectural timing side channels in processors.
  - SIGFuzz discovered 3 new side channels in Rocket and BOOM processors.
  - SIGFuzz is open-sourced: https://github.com/bu-icsg/SIGFuzz

References

1. S. Canakci, C. Rajapaksha, A. Nataraja, L. Delshadtehrani, M. Taylor, M. Egele and A. Joshi, “ProcessorFuzz: Processor Fuzzing with Control and Status Registers Guidance,” to appear in Proc. IEEE International Symposium on Hardware Oriented Security and Trust (HOST) 2023.
2. C. Rajapaksha, L. Delshadtehrani, M. Egele and A. Joshi, “SIGFuzz: A Framework for Discovering Microarchitectural Timing Side Channels,” to appear in Proc. Design, Automation and Test in Europe (DATE) 2023.
3. Google, “Oss-fuzz: Continuous fuzzing for open source software,” https://github.com/google/oss-fuzz, 2016.
4. Google, “American fuzzy lop,” https://github.com/google/AFL, 2017.
5. K. Laeufer, J. Koenig, D. Kim, J. Bachrach, and K. Sen, “Rfuzz: Coverage-directed fuzz testing of rtl on fpgas,” in International Conference on Computer-Aided Design, 2018, pp. 1–8.
6. S. Canakci, L. Delshadtehrani, F. Eris, M. B. Taylor, M. Egele, and A. Joshi, “Directfuzz: Automated test generation for rtl designs using directed graybox fuzzing,” in Design Automation Conference, 2021.
7. J. Hur, S. Song, D. Kwon, E. Baek, J. Kim, and B. Lee, “Difuzzrtl: Differential fuzz testing to find cpu bugs,” in Security and Privacy, 2021, pp. 1286–1303.
8. K. Asanovic´ et al., “The rocket chip generator,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17, 2016.
9. C. Celio, D. A. Patterson, and K. Asanovic, “The berkeley out-of-order machine (boom): An industry-competitive, synthesizable, parameterized risc-v processor,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2015-167, Jun 2015.
10. D. Petrisko, F. Gilani, M. Wyse, D. C. Jung, S. Davidson, P. Gao, C. Zhao, Z. Azad, S. Canakci, B. Veluri, T. Guarino, A. Joshi, M. Oskin, and M. B. Taylor, “Blackparrot: An agile open-source risc-v multicore for accelerator socs,” IEEE Micro, vol. 40, no. 4, pp. 93–102, 2020.