Accelerating Coverage-based Greybox Fuzzing to Rapidly Identify Bugs in Processors

Brennen Ho[1,2], Chathura Rajapaksha[2], Ajay Joshi[2]
Palo Alto High School, 50 Embarcadero Rd, Palo Alto CA[1]; Boston University, Boston MA[2]
brennen.ho@gmail.com

Introduction

Fuzzing

- A popular method of identifying bugs in software.
- Test a program repeatedly with a pool of random inputs to trigger bugs.
- Coverage-based greybox fuzzing is a prominent fuzzing technique[1] that uses coverage from the execution to guide the generation of inputs
- Fuzzing has recently been adopted for hardware testing[2], but the effectiveness of existing frameworks is limited due to the coverage metrics used.

ProcessorFuzz (PF)

- A hardware fuzzing framework for testing processors and addresses limitations by using a new coverage metric based on control and status registers (CSRs), which store processors' internal state.

To assist in further research with PF, this project investigates methods to accelerate PF so more test iterations can be run in a given time, thereby increasing the chances of finding bugs.

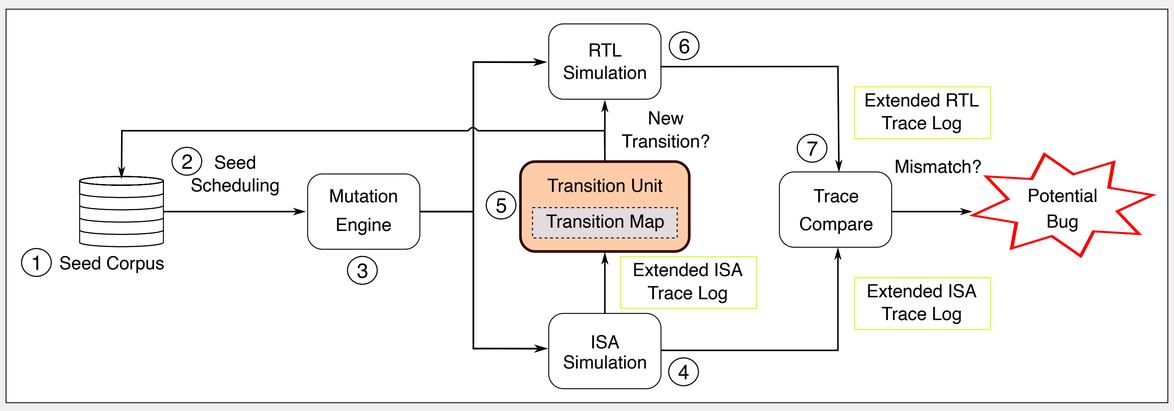

Figure 1: PF logic diagram outlining fuzzing steps, including use of transition map to utilize coverage-based greybox fuzzing and steps to identify potential bugs[3].

Methods

Code profiling was used to determine the most time-consuming operations. Python's cProfile library, a C extension with minimal overhead, tracked the runtime of all functions called during the fuzzing process.

Functions	Time (s)
trace_compare (utils.py)	2.777
read_spike_trace (spike_log_to_trace_csv.py)	8.068
read_spike_instr (spike_log_to_trace_csv.py)	1.446
extract_transitions (utils.py)	1.266

Table 1: The total time functions and all of their sub-routines took to execute. Functions from standard python libraries were ignored, and the test was run for 100 fuzzing iterations.

The time-consuming functions were evaluated to identify possible optimizations.

- Compression: compress the comparison of each line of the logs
- Data Structure Choice: utilize more efficient data structures that execute in C instead of python
- CSV Conversion Avoidance: read values directly from trace log

Results

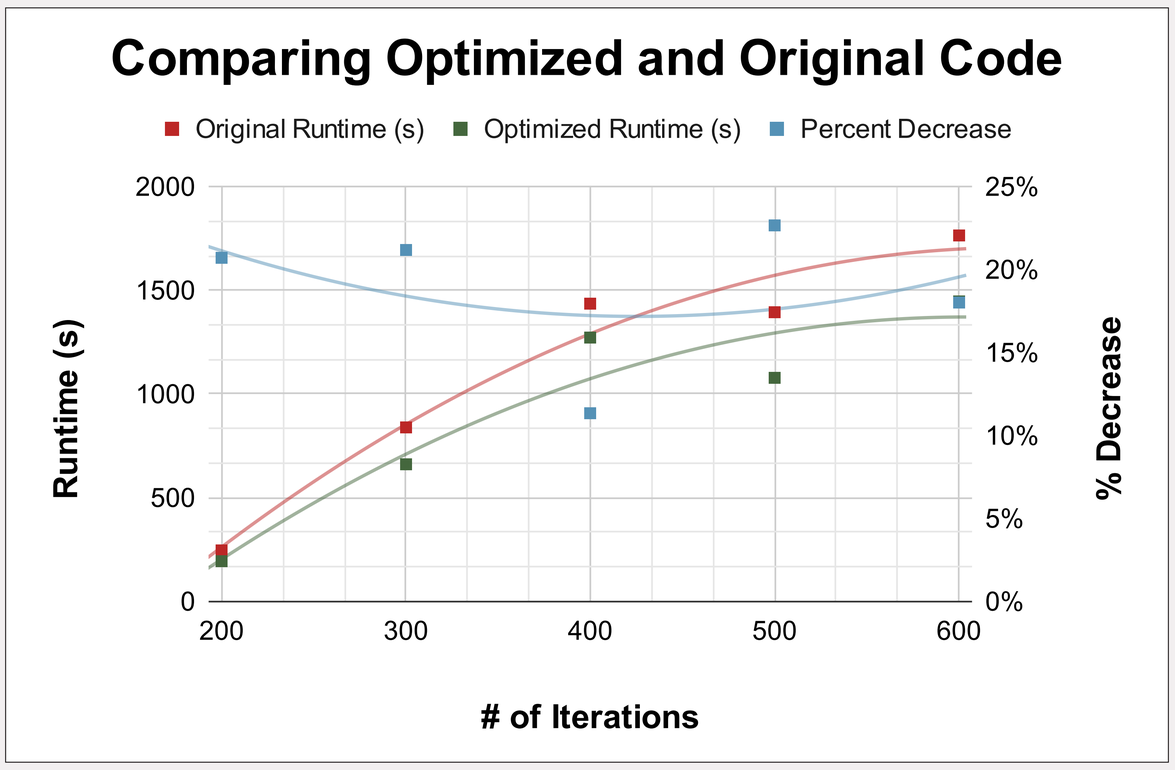

Figure 2: Runtimes for original and optimized code, as well as time reduction between the two implementations.

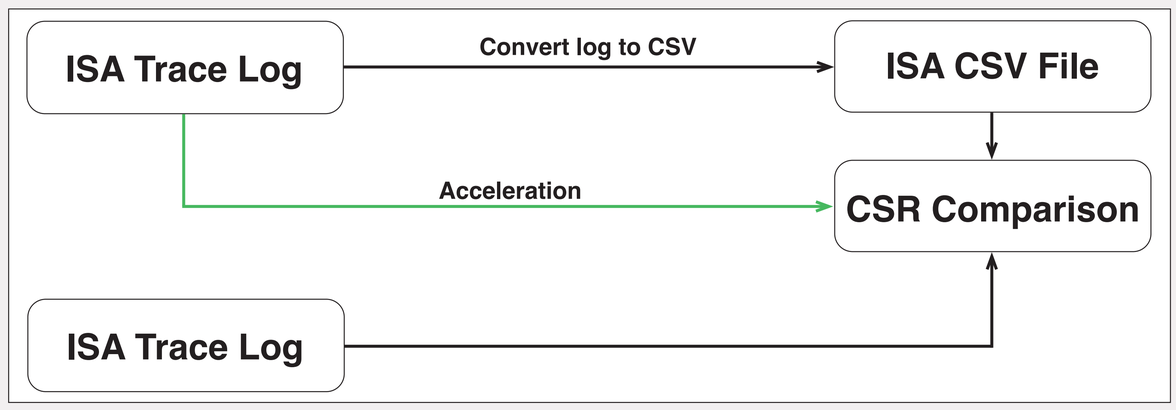

Figure 3: Flow diagram for trace_compare function outlining comparison steps and omitted CSV conversion.

Discussion

- By removing the CSV conversion and more efficiently reading CSR values directly from the ISA trace log, there was an 18.97% average reduction in runtime in five experiments with iteration counts ranging from 200 to 600.
  - A 20% improvement will be beneficial in lengthy sessions during which tens of thousands of iterations are run at once.
  - Fuzzing operations will require fewer resources, as files do not need to be generated and removed.
- Since ISA and RTL simulations of a particular test are independent of other tests, parallelization can be used to accelerate the fuzzing process.
  - Multi-threading can be used to run several instances of ISA and RTL simulations in parallel, which can significantly reduce the bug-finding time.

References

1. Google. 2016. OSS-Fuzz: Continuous Fuzzing for Open Source Software. https://github.com/google/oss-fuzz
2. J. Hur, S. Song, D. Kwon, E. Baek, J. Kim, and B. Lee. 2021. DiFuzzRTL: Differential Fuzz Testing to Find CPU Bugs. In 2021 2021 IEEE Symposium on Security and Privacy (SP). IEEE Computer Society, Los Alamitos, CA, USA, 1286–1303. https://doi.org/10.1109/SP40001.2021.00103
3. Canakci. 2022. Directing Greybox Fuzzing to Discover Bugs in Hardware and Software. https://open.bu.edu/handle/2144/44702

Acknowledgments

I would like to thank Chathura Rajapaksha for his constant support and mentorship throughout this project and Professor Joshi for giving me the opportunity to complete this project as part of the Integrated Circuits and Systems Group. I would also like to thank Boston University's Research in Science and Engineering (RISE) program for this enriching and unique opportunity.

BOSTON UNIVERSITY