RtlFuzzLab: Building A Modular Open-Source Hardware Fuzzing Framework

Brandon Fajardo, Kevin Laeufer, Jonathan Bachrach, and Koushik Sen
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley, USA
{brfajardo, laeufer, jrb, ksen}@berkeley.edu

Coverage directed mutational fuzz testing has become a popular tool for automated software testing and bug detection. This technique works by providing inputs to a program and measuring which branches in the program under test are explored during execution of those seed inputs. The fuzzer then automatically mutates the original inputs by applying a series of deterministic and non-deterministic mutation operations. The program under test is ran on these newly generated mutant inputs. A new input will be retained to serve as the start for new mutations if and only if it produces coverage of branches which have not been seen before or significantly increases the times a branch has been covered. The vast majority of mutant inputs are immediately discarded as they do not improve the coverage. This automated fuzzing has been very successful in uncovering countless software bugs, however, it has yet to catch on for testing digital hardware designs.

The initial work on applying the fuzzing algorithm used for software testing to register transfer level (RTL) designs described the various added degrees of freedom that we face in the hardware context [1]. One important decision is how to take the sequence of bytes provided by a software fuzzer and use it in a testbench. The simplest solution is to just apply new data to the input wires of the design under test and to stop the test execution once we run out of data provided by the fuzzer. More recent work has proposed to instead covert the raw bytes into read or write tile link transactions through the use of a grammar [2]. Another question is what coverage metric to use as feedback to the fuzzer. The original paper counted the number of toggles on a multiplexer control signal [1], while subsequent papers have suggested to only count full toggles [3] or to use the branch coverage of a software simulation model as proxy for line coverage of the RTL description [2]. The choice of initial input seeds to begin fuzzing with can also affect the results.

We present our ongoing work on RtlFuzzLab, an open-source framework based on the FIRRTL RTL compiler infrastructure [4], that makes it easy to explore new fuzzing ideas. We provide an easy integration of the popular AFL fuzzer and are working on adding our own fuzzer implementation to be able to experiment with custom mutations and fuzz scheduler designs. Our direct harness is based on the ideas from the Rfuzz paper [1] and works on any RTL circuit with a single input clock and reset. It can be easily switched out for a bus-centric harness which re-implements the tile link specific harness from the work on Fuzzing hardware like software [2]. We provide various coverage metrics like mux toggle coverage [1], full mux toggle coverage [3] and HDL line coverage (approximating the coverage used in [2]). These coverage metrics can be combined with post-processing functions and arbitrary combinations of them can be selected as feedback to the fuzzer. We also include some simple benchmarks as well as scripts to run benchmarks and analyze results, making it easy to prototype new fuzzing ideas. The code is open source under a BSD license and available on GitHub: https://github.com/ekiwi/rtl-fuzz-lab

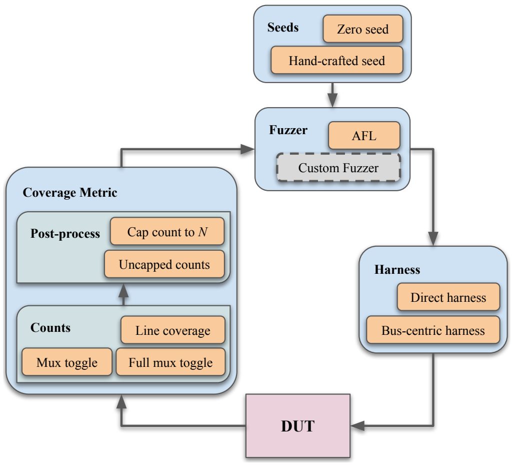

Fig. 1. Software framework visualization

This work was supported in part by Semiconductor Research Corporation.

References

[1] K. Laeufer, J. Koenig, D. Kim, J. Bachrach, and K. Sen, "Rfuzz: Coverage-directed fuzz testing of rtl on fpgas," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2018, pp. 1–8.

[2] T. Trippel, K. G. Shin, A. Chernyakhovsky, G. Kelly, D. Rizzo, and M. Hicks, "Fuzzing hardware like software," *arXiv preprint arXiv:2102.02308*, 2021.

[3] T. Li, H. Zou, D. Luo, and W. Qu, "Symbolic simulation enhanced coverage-directed fuzz testing of rtl design," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2021, pp. 1–5.

[4] A. Izraelevitz, J. Koenig, P. Li, R. Lin, A. Wang, A. Magyar, D. Kim, C. Schmidt, C. Markley, J. Lawson *et al.*, "Reusability is FIRRTL Ground: Hardware Construction Languages, Compiler Frameworks, and Transformations," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2017, pp. 209–216.