HWFuzz: An FPGA-Accelerated FuzzingFramework for Efficient RISC-V Verification

Yang Zhong[1,2], Haoran Wu[3*], Yungang Bao[1,2] and Kan Shi[1,2]

[1] State Key Lab of Processors, Institute of Computing Technology, Chinese Academy of Sciences
[2] University of Chinese Academy of Sciences
[3] University of Cambridge

Vision & Value

- To address the limitations of existing DV frameworks, hardware fuzzing has emerged as a promising approach, inspired by its widespread use in the software testing domain.
- Recent advancements in hardware fuzzing have led to the discovery of a significant number of bugs in open-source RISC-V processor cores, such as Rocket Core, BOOM, and CVA6, further demonstrating its practical effectiveness in real-world scenarios.

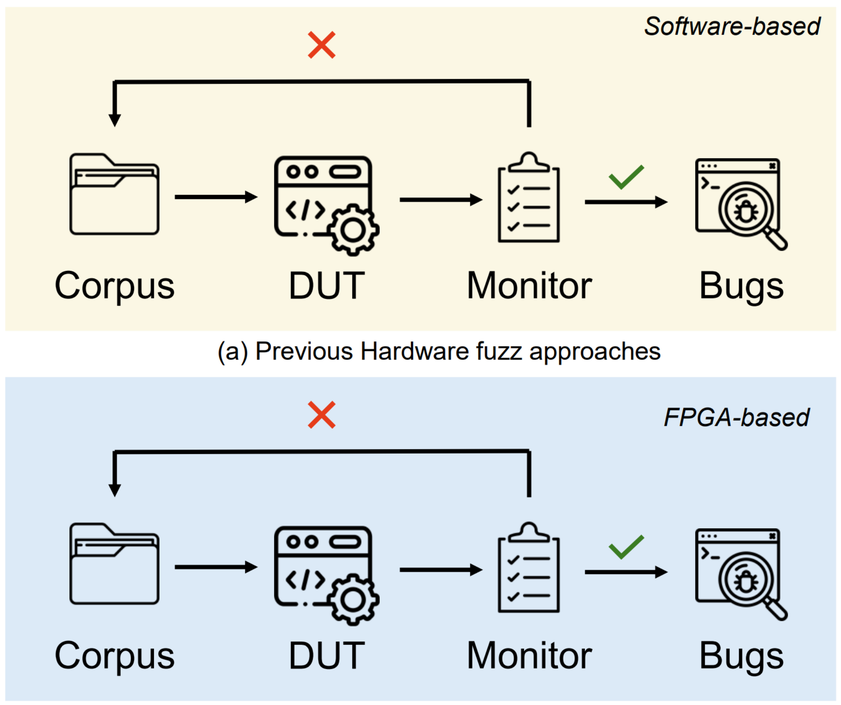

Figure 1: Comparison between the previous Hardware Fuzzing approaches and the proposed approach

Previous methods in Hardware Fuzzing

Software-based

- Test stimulus generation is slow due to limited software performance.
- DUT execution becomes a bottleneck because of time-consuming simulations.

Offload DUT to FPGA

- Significantly improves DUT execution speed.
- New bottlenecks emerge:Test stimulus generation still remains slow . Communication latency between the host and the FPGA limits overall fuzzing throughput.

Our solution：

- Developed a synthesizable and highly configurable hardware fuzzer IP.
- Implements a fully automated hardware fuzzing-verification loop entirely on FPGA
- Detects potential vulnerabilities by comparing DUT execution with a software reference model.

The overall architecture

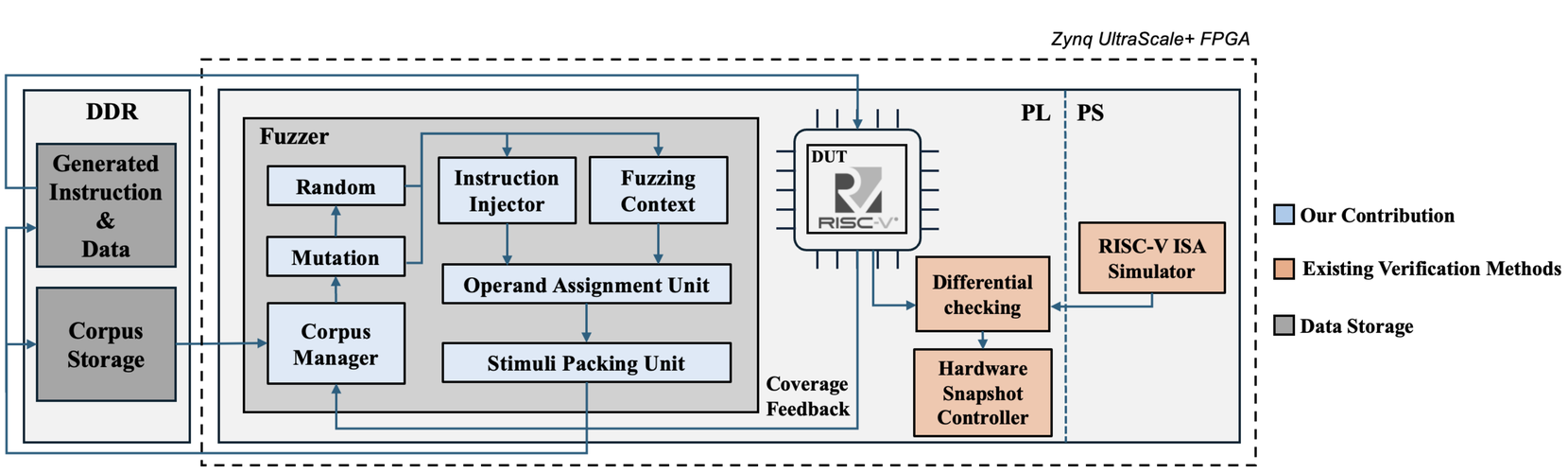

Figure 2: The overall architecture of our proposed method.

Stimuli Constraints

- Implement hardware-level constraints on control-flow instruction jump ranges.
- Constraints significantly increase the proportion of generated instructions that can be executed, enhancing overall coverage.

Coverage-Directed Generation

- Two Fuzzing Modes:
  - Random Mode: Generates instructions purely at random.
  - Mutation Mode: Adjusts operands and context of previously generated stimuli.
- Corpus-Guided Mutation:
  - Stores stimuli and corresponding DUT coverage in a corpus.
  - Selectively mutates high-coverage stimuli to maximize DUT coverage and improve verification efficiency.

Stimuli Packing.

- Generates both instructions and data, stored in DDR.
- Random data values are created to support Load/Store instructions, and ensuring memory-access operations have valid operands.
- A multi-stage pipeline refines raw opcodes into executable instructions by adding context, helper instructions, and operands.

Evaluation

Platform: Fidus Sidewinder board

- with a Xilinx Zynq UltraScale+ XCZU19EG FPGA and two 16GB DDR4 memories.

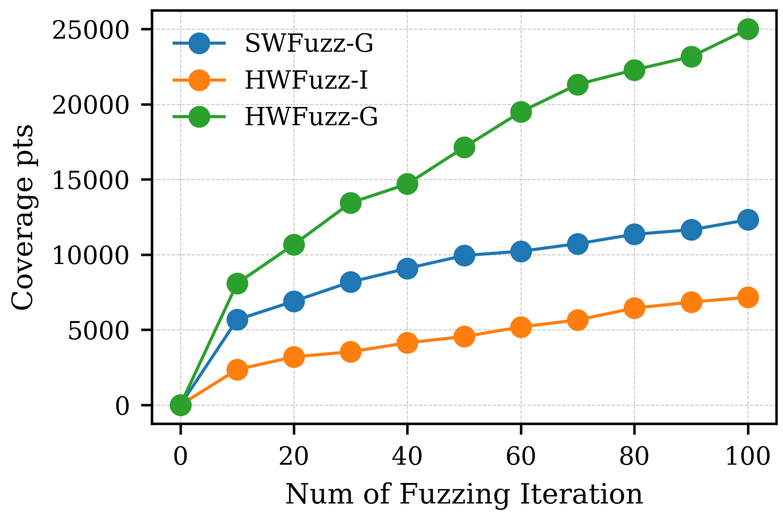

Figure 3: Coverage comparison

Table 1: *Resource Usages*

Resource	Fuzzer	Infra	DUT
LUTs	68377 (13.08%)	8217 (1.57%)	209865 (40.15%)
Block RAMs	192.5 (19.56%)	325 (33.03%)	20 (2.03%)
Registers	92407 (8.84%)	12749 (1.22%)	118683 (11.35%)