AccUnit: Accelerating Unit Level Verification for RISC-V Processors Using FPGA

Chenang Zhu[1,2*] Weidong Li[3*†] Yungang Bao[1,2] 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]ShanghaiTech University

Abstract

Verification consumes up to 70% of the entire chip development cycle, creating a critical efficiency bottleneck—especially when developing complex chips such as RISC-V processors. However, current research on agile verification primarily focuses on system-level verification. As chip development progresses, the cost of fixing bugs may increase exponentially. To enable a shift-left verification effort, we propose AccUnit: an end-to-end tool flow for efficient unit-level verification of RISC-V processors using heterogeneous hardware acceleration. AccUnit automatically duplicates the design-under-test (DUT) to fit the FPGA alongside software reference models running on the host, while generating distinct input stimulus sets for each DUT and reference model pair. The proposed system monitors and compares key I/O ports between the DUT and reference model, enabling massive parallel execution of automated unit-level self-checking for RISC-V processor modules. We also integrate a coverage collection scheme where synthesizable coverpoints are automatically created and inserted into the DUT during compile time. This allows coverage information to be collected on the FPGA during operation, enabling comprehensive evaluation of functional verification quality. The results demonstrate that our approach achieves up to a 75× performance improvement compared to software simulation using Verilator.

Introduction

As computer architecture and hardware design become increasingly complex, verification has emerged as the most critical and time-consuming phase of chip development[1]. Unit testing verifies at the module level, helping verification engineers identify design defects early and prevent their spread to later development stages. In this work, we propose AccUnit, an end-to-end FPGA-accelerated verification tool flow specifically for unit-level verification of RISC-V processors, aiming to shift key verification processes earlier in the development flow.

AccUnit highlights the following contributions: **(1) Parallel Verification**: AccUnit automatically duplicates the DUT to fit the entire FPGA, together with the corresponding stimulus generation and reference model running on the host, allowing massive parallel execution of unit-level verification runs. **(2) Synthesizable Coverpoints**: AccUnit automatically instruments the DUT with synthesizable coverpoints[2], enabling the collection of various code coverage metrics on the FPGA, including line coverage, ready-valid coverage, and toggle coverage. **(3) Automatic Deployment**: AccUnit provides a fully automated, end-to-end verification workflow. It seamlessly integrates multiple copies of `<DUT, reference model>` pairs into the heterogeneous system for parallel verification.

The AccUnit Architecture

The overall architecture of AccUnit is shown in Figure 1. AccUnit harnesses the inherent parallelism of FPGA and the multi-core capabilities of the host CPU by executing multiple copies of `<DUT, reference model>` pair in parallel to maximize verification efficiency. Synthesizable coverpoints are inserted into the DUT during compilation for coverage collection, and the coverage results are transferred back to the host for merging and analysis. The entire workflow is fully automated.

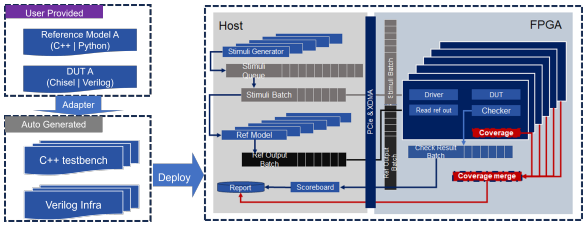

Figure 1: *Overall architecture of AccUnit*

Parallel Verification AccUnit leverages both FPGA parallelism and host CPU multi-core capabilities by running multiple `<DUT, reference model>` pairs in parallel with different input patterns, maximizing verification efficiency. During compilation, synthesizable coverpoints are inserted into the DUT to collect coverage data during runtime, which is then transferred to the host for merging and analysis. This

*Equal contribution.
†Weidong Li finishes this work during internship at Institute of Computing Technology, Chinese Academy of Sciences.
‡Corresponding author: `shikan@ict.ac.cn`

entire process runs automatically. Stimuli Generators and reference models run on the host, implemented in threads to enable efficient communication through shared memory. To mitigate the performance overhead from frequent I/O operations, data batching is used to aggregate individually generated stimuli into a single batch, thereby reducing the of I/O operations.

A direct one-to-one execution between stimuli generation and reference model execution can be inefficient due to potential speed mismatches. To solve this, we implement stimuli queues that decouple these two processes. The system works by pushing generated stimuli into a queue until it reaches a predefined batch size n. These n stimuli are then dequeued and combined into a single batch. This batched data is transferred to the FPGA in one I/O operation for distribution to each DUT, while simultaneously being sent to each reference model on the host.

To optimize CPU utilization and maintain balanced execution, AccUnit dynamically adjusts stimuli generator thread counts based on the relative speeds of the generator and reference model. When the stimuli generator is m times faster than the reference model, AccUnit creates $m \times n$ reference model threads paired with n stimuli generation threads, and vice versa.

Synthesizable Coverpoints Coverage provides confidence that a design performs as expected and indicates when verification can be considered complete. Traditional software-based coverage collection suffers from poor performance and is unsuitable for FPGA implementations. AccUnit solves this by integrating a synthesizable coverage collection tool[2] that captures data directly from the FPGA-implemented DUT. Using the FIRRTL compiler, coverpoints are inserted into the RTL design as counters. These counters accumulate data during runtime, enabling real-time coverage collection and analysis.

Automatic Deployment AccUnit automates the process of adapting drivers and monitors for DUTs and their reference models, duplicating stimulus generators and `<DUT, reference model>` pairs, inserting coverpoints into DUTs, and integrating all components into the testbench. Users need only provide the DUT and reference model files, ensuring both have identical ports and control logic. The system drivers and monitors act as adapters, converting different module ports into the unified interface and making it adaptable to various `<DUT, reference model>` pairs. AccUnit determines the maximum supported parallelism based on post-synthesis resource utilization and allows users to specify their desired parallelism level. With this parameter set, AccUnit automatically duplicates the DUT and reference model instances, along with their supporting infrastructure, to build the testbench.

Evaluation and Results

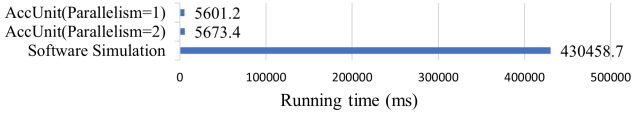

Figure 2: *Performance comparison between AccUnit and software simulation (1,000,000 iterations)*

AccUnit was implemented and evaluated on a Virtex UltraScale+ VCU128 FPGA board using the Vector module from the Xiangshan RISC-V core as the DUT and its corresponding reference model. Figure 2 shows the running time for one verification iteration. AccUnit achieves up to a 75× performance improvement compared to software simulation using Verilator. When running two `<DUT, reference model>` pairs in parallel, the execution time per iteration remains nearly constant, while the number of stimuli processed per iteration doubles—effectively doubling the overall verification throughput.

Table 1: *Resource Utilization of Coverage Instrumentation*

Module	Cov Count	LUTs	Registers	LUTs with Cov	Registers with Cov
RAS	73	3043 (0.23%)	2007 (0.08%)	3206 (0.25%)	2072 (0.08%)
FTB	629	3167 (0.24%)	2621 (0.10%)	3697 (0.28%)	3175 (0.12%)
Vector	7078	152727 (11.72%)	28143 (1.08%)	168721 (12.94%)	35184 (1.35%)

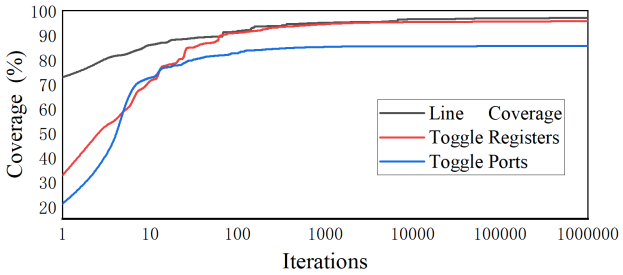

Figure 3: *Coverage of Vector Module Collected by AccUnit*

Figure 3 shows the growth of various code coverage metrics for the Vector module across iterations. All coverage metrics converge after thousands of iterations, which is influenced by the DUT size and verification throughput. Table 1 shows the resource utilization for coverage instrumentation across various modules using AccUnit, showing that AccUnit enables real-time coverage collection on the FPGA to assess verification progress while maintaining acceptable hardware resource overhead.

References

[1] Wen Chen et al. "Challenges and trends in modern SoC design verification". In: *IEEE Design & Test* 34.5 (2017), pp. 7–22.

[2] Kevin Laeufer et al. "Simulator Independent Coverage for RTL Hardware Languages". In: *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*. ASPLOS 2023. Association for Computing Machinery, 2023, pp. 606–615. DOI: 10.1145/3582016.3582019.