Toffee: an Efficient and Flexible Python Testing Framework for Chip Verification

Jincheng Liu[1], Zhicheng Yao[1], Yunlong Xie[1], Zechen Yang[2], Junyue Wang[1], Xiao Chen[1], Kan Shi[1], Sa Wang[1] and Yungang Bao[1]

[1] Institute of Computing Technology, Chinese Academy of Sciences, [2] University of Chinese Academy of Sciences

Abstract

Functional verification can take up to 60% of IC/ASIC development time, yet remains poorly integrated with modern software tools. We propose Toffee a Python-based framework that improves efficiency through:

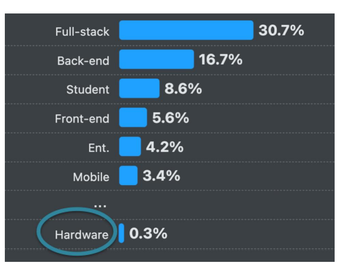
(1) async function modeling of hardware APIs, (2) hook-enabled reference models, and (3) test-driven execution.

Results show up to 86% fewer lines of code, 90% faster execution, and setup time under 10 hours, enabling seamless integration with software workflows.

Huge resource differences in the fields of software and hardware

Advantages of Software Testing Practices

Tools like pytest enable **auto test discovery and agile iteration,** greatly improving flexibility and efficiency.



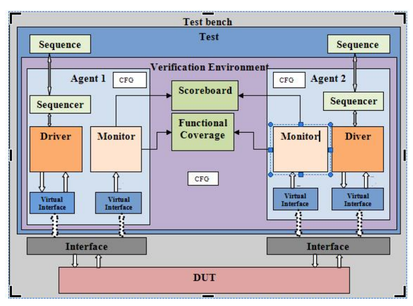
Developer Base Advantage

Software has a much **larger developer pool**, driving faster tool evolution and broader adoption.

Current challenges in integrating with the software ecosystem

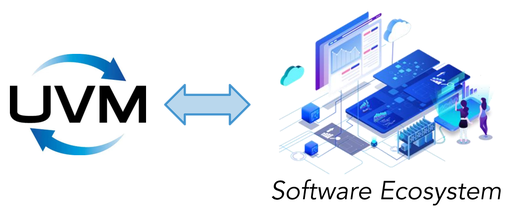
Low Coding Efficiency

Legacy structures remain **complex**, slowing iteration.



Poor Tool Integration

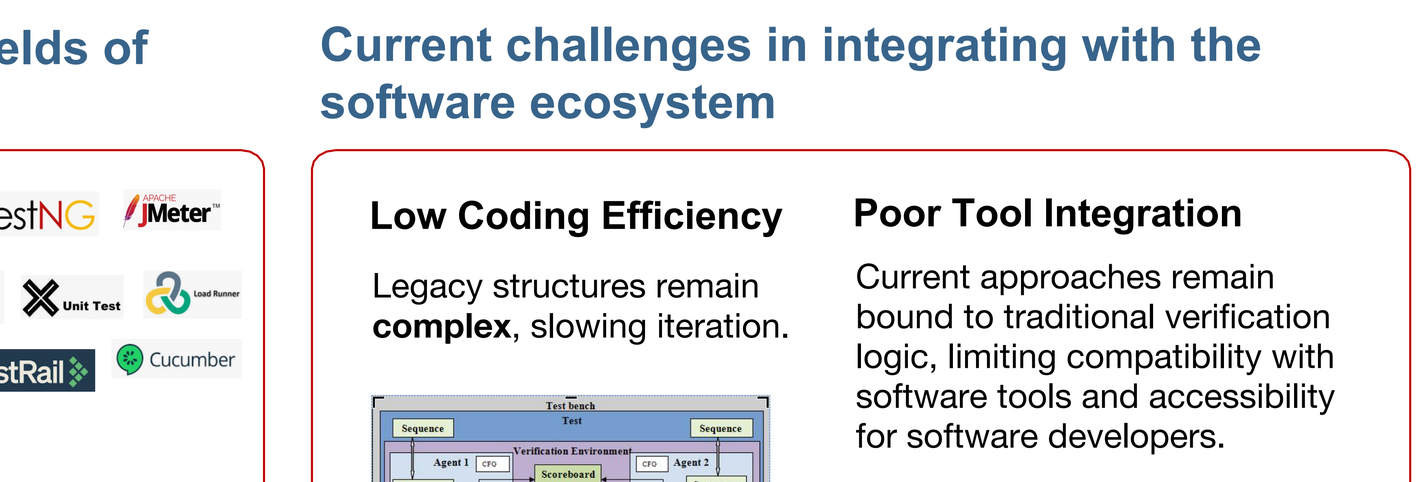
Current approaches remain bound to traditional verification logic, limiting compatibility with software tools and accessibility for software developers.



Software Ecosystem

Toffee enables fast, software-friendly hardware verification.

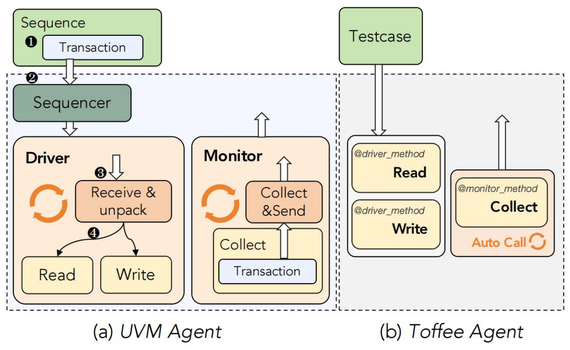
Toffee Architecture



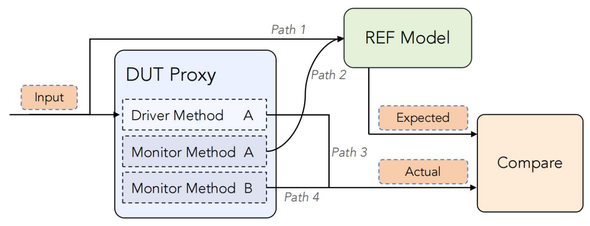
DUT Proxy: High-Level Abstraction for DUT Access

- Wraps low-level pin/clock control into API-style async functions.
- Driver and monitor methods act as the DUT's software-facing API.
- Optional features (e.g., sequencer, custom transaction) added only when needed.

Enables fast, modular, and developer-friendly verification



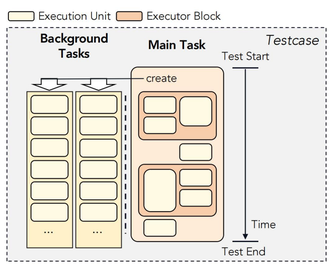
Model Hub: Simplified and Flexible Integration



- Toffee connects the reference model using four hook types: driver hook, monitor hook, driver port, and monitor port.
- Hooks provide clean and structured connections to DUT inputs and outputs.
- Supports all data paths—from test case or DUT to reference model and back.
- Enables automatic result comparison, reducing manual effort.

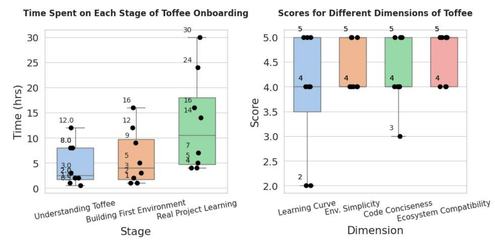
Test Scenario Layer: Software-Friendly, Test-Driven Execution

- Toffee uses a push model—tests explicitly control execution.
- Test cases are async functions, fully managed by software frameworks.



Results and Conclusion

Results



- **Efficiency**: Toffee reduces LOC by up to 86.31%, showing strong productivity in both small and large designs.
- **Usability**: Software developers set up environments in 1.8–9.8 hours.
- **Integration**: Works seamlessly with tools like pytest and hypothesis.

Conclusion

- Toffee bridges hardware and software verification with async modeling, hooks, and test-driven execution.
- It reduces code by up to 86.41% and integrates smoothly with software tools.
- Real cases and user feedback confirm its efficiency and practical value.