Software-Hardware Co-Verification for Traditional Verification Frameworks

Fangyuan Song[1,*], Yunlong xie[1], Jincheng Liu[1]

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

Abstract

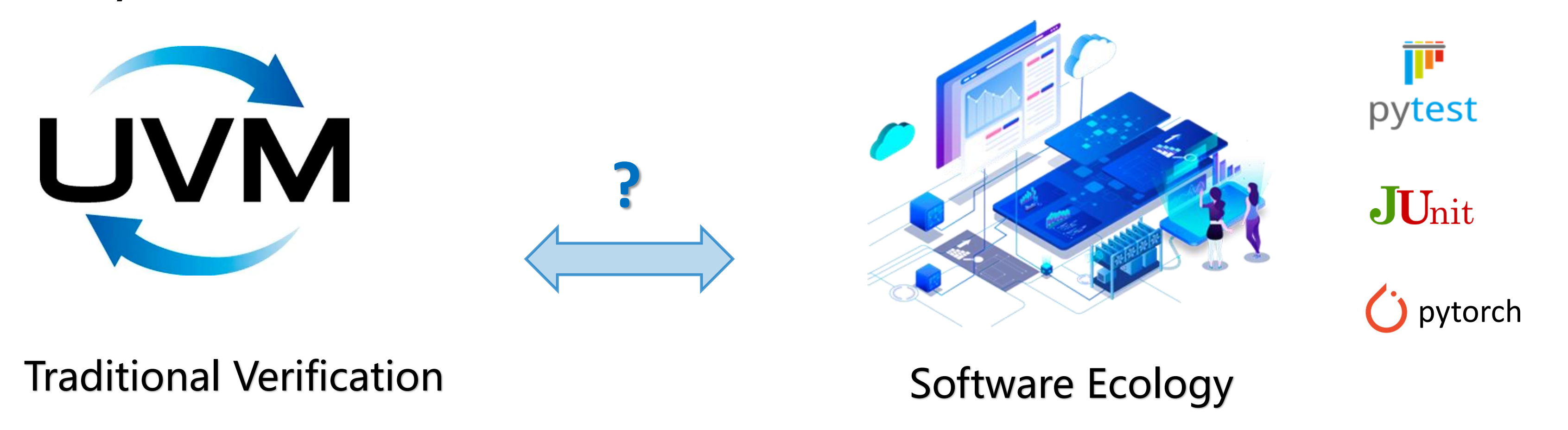
Chip verification is a critical phase in the design process, constitutes up to 70% of the total development cycle. To meet escalating computational demands and diversifying application scenarios, agile design methodologies and high-level abstraction languages have substantially improved design efficiency; however, verification efficiency has lagged behind. While high-level languages (HLLs) offer novel approaches through their concise syntax and rich ecosystems, traditional hardware description language (HDL)-based verification frameworks remain dominant, leading to challenges such as incompatibility between frameworks like Cocotb and legacy verification IPs, insufficient standardized multi-language support for cross-language calls (e.g., Python), and exacerbated co-verification complexity due to fundamental disparities in software-hardware programming paradigms. This study introduces T-SHCV, the first multi-language software-hardware co-verification framework, which addresses these challenges through hardware-friendly communication protocols for efficient cross-language data transmission, software-interface-based hardware abstraction enabling software-driven control of traditional hardware verification frameworks, and software-package-based extensions providing unified programming interfaces for diverse languages. Experimental results demonstrate that T-SHCV maintains performance comparable to DPI-C while reducing connection code requirements by 86% for multi-language reference model integration, thus bridging traditional verification environments with modern software toolchains and establishing a new co-verification paradigm.

INTRODUCTION

Motivation & Challenge

1. Bridging Software and Hardware Verification

- By integrating software-driven verification methodologies to overcome the constraints of traditional hardware validation, we achieve exponential improvements in verification efficiency.



Traditional Verification

Software Ecology

Challenge 1

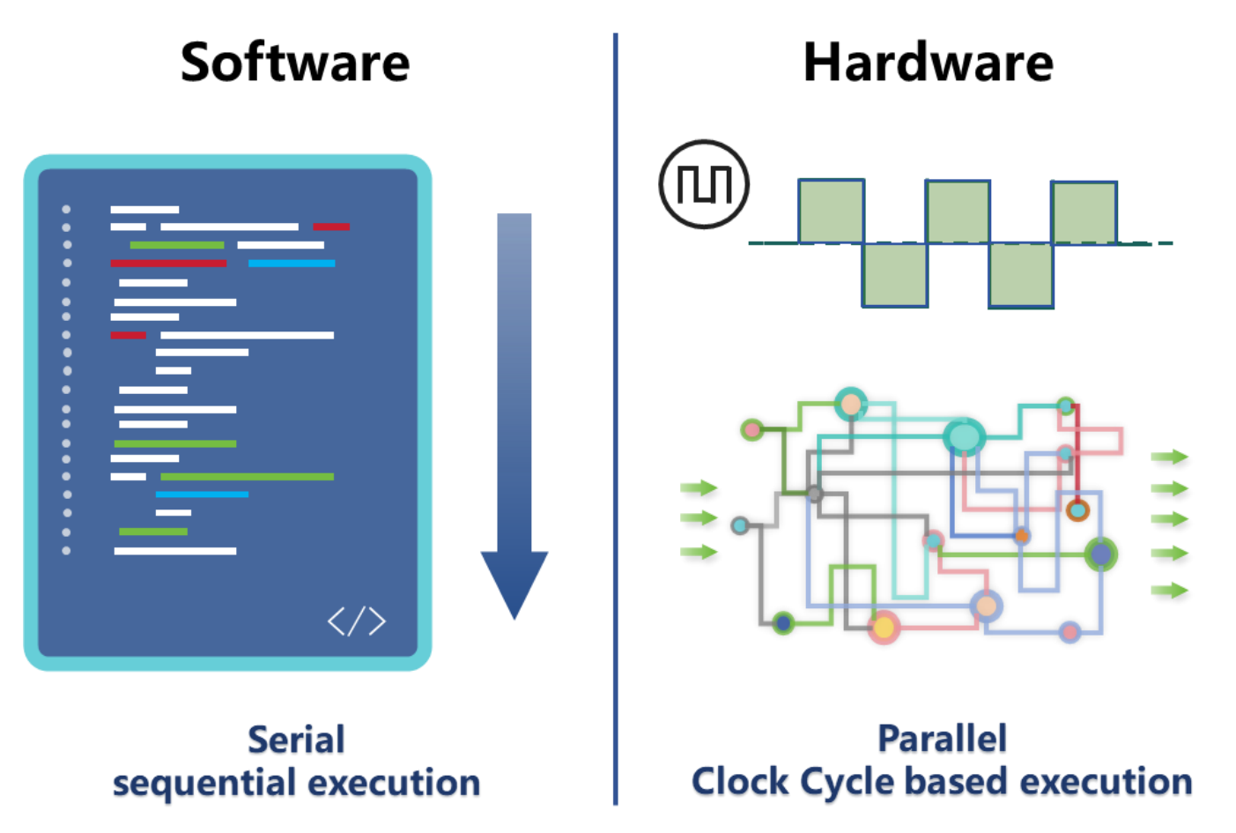
Fundamental disparities in programming create bidirectional communication barriers.

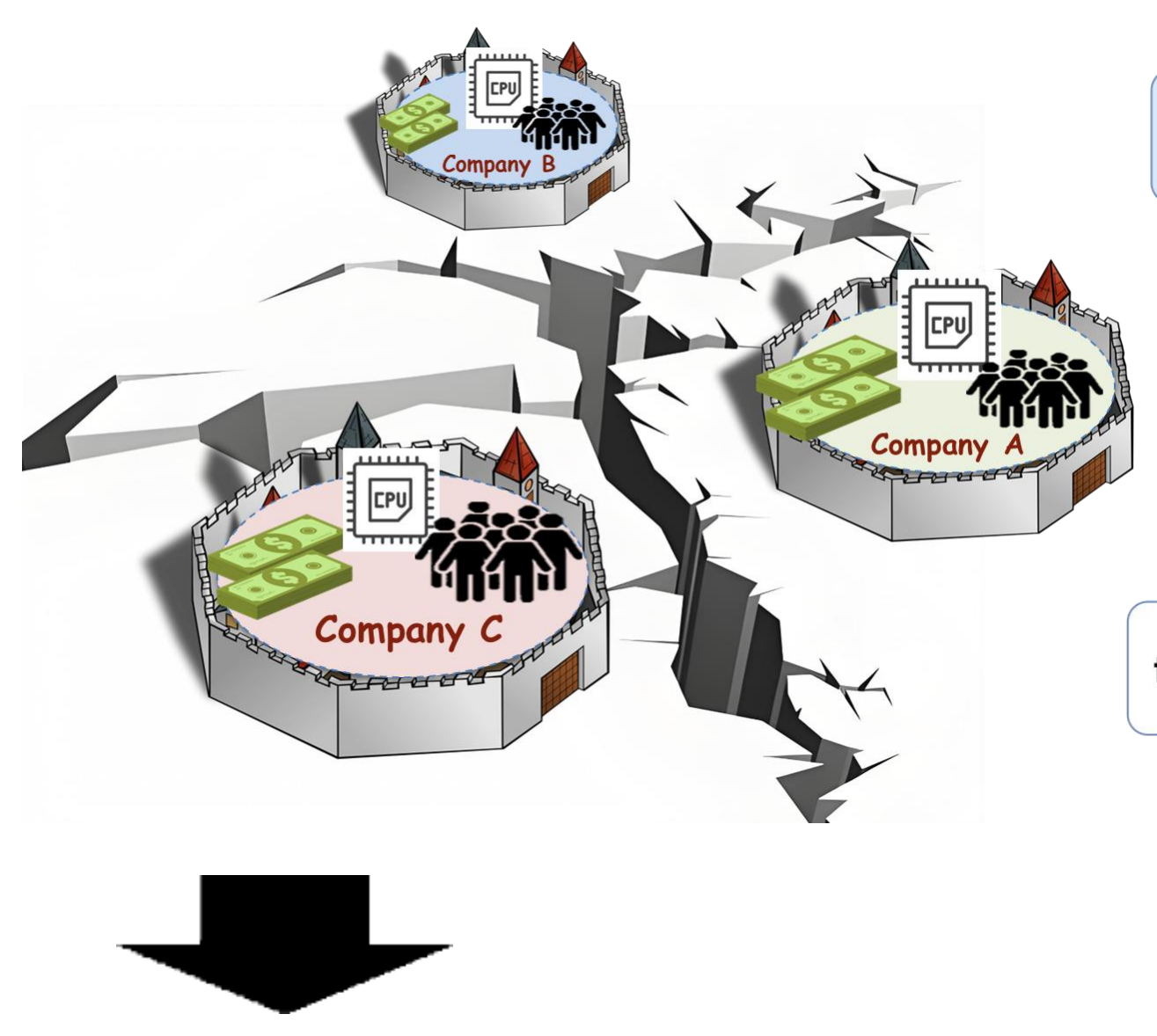
Challenge 2

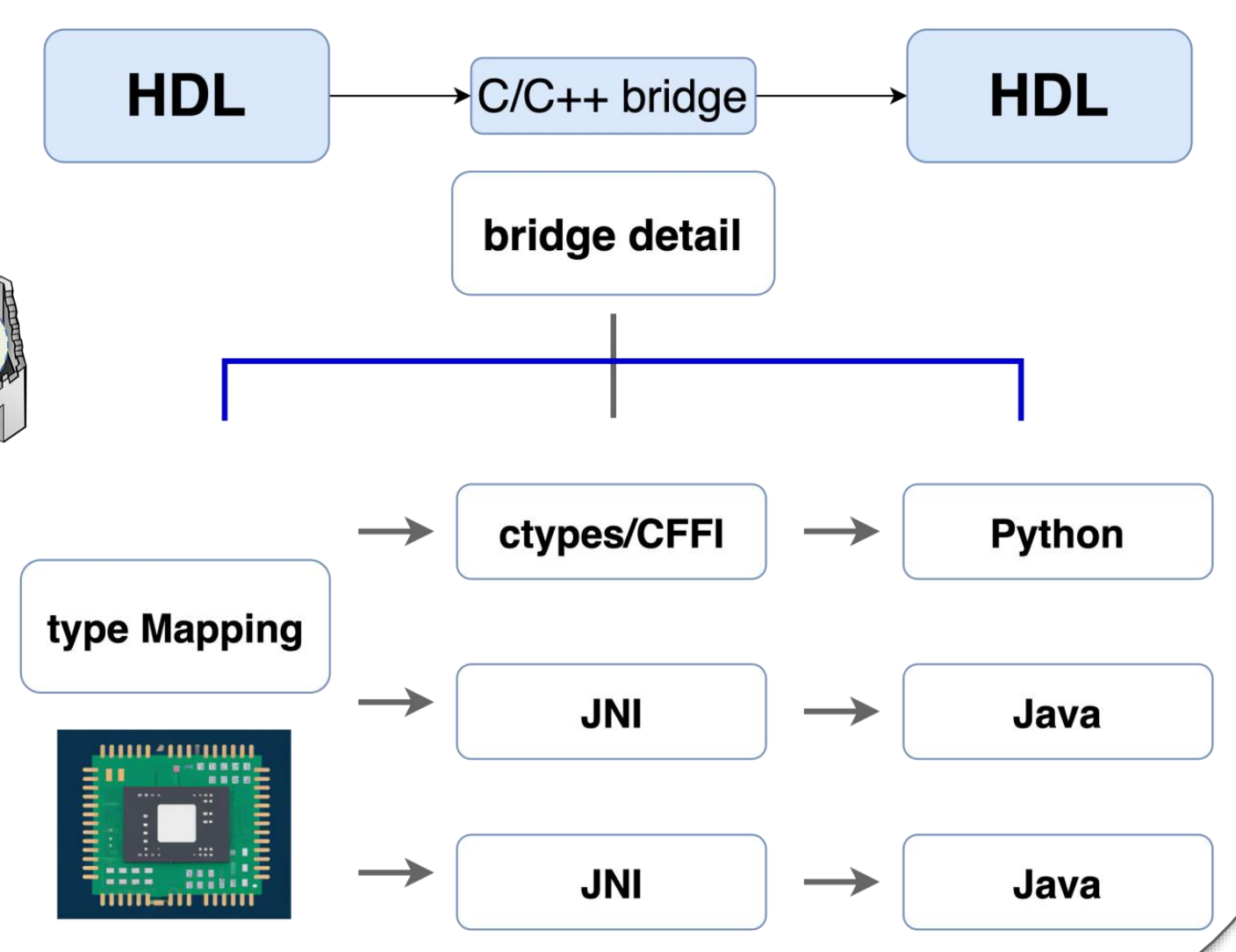
Legacy integration methods rely on cumbersome signal-level operations rather than modern transaction

Challenge 3

Language-specific interface implementations require disparate adapters for each HLL-HDL pairing, increasing maintenance overhead.



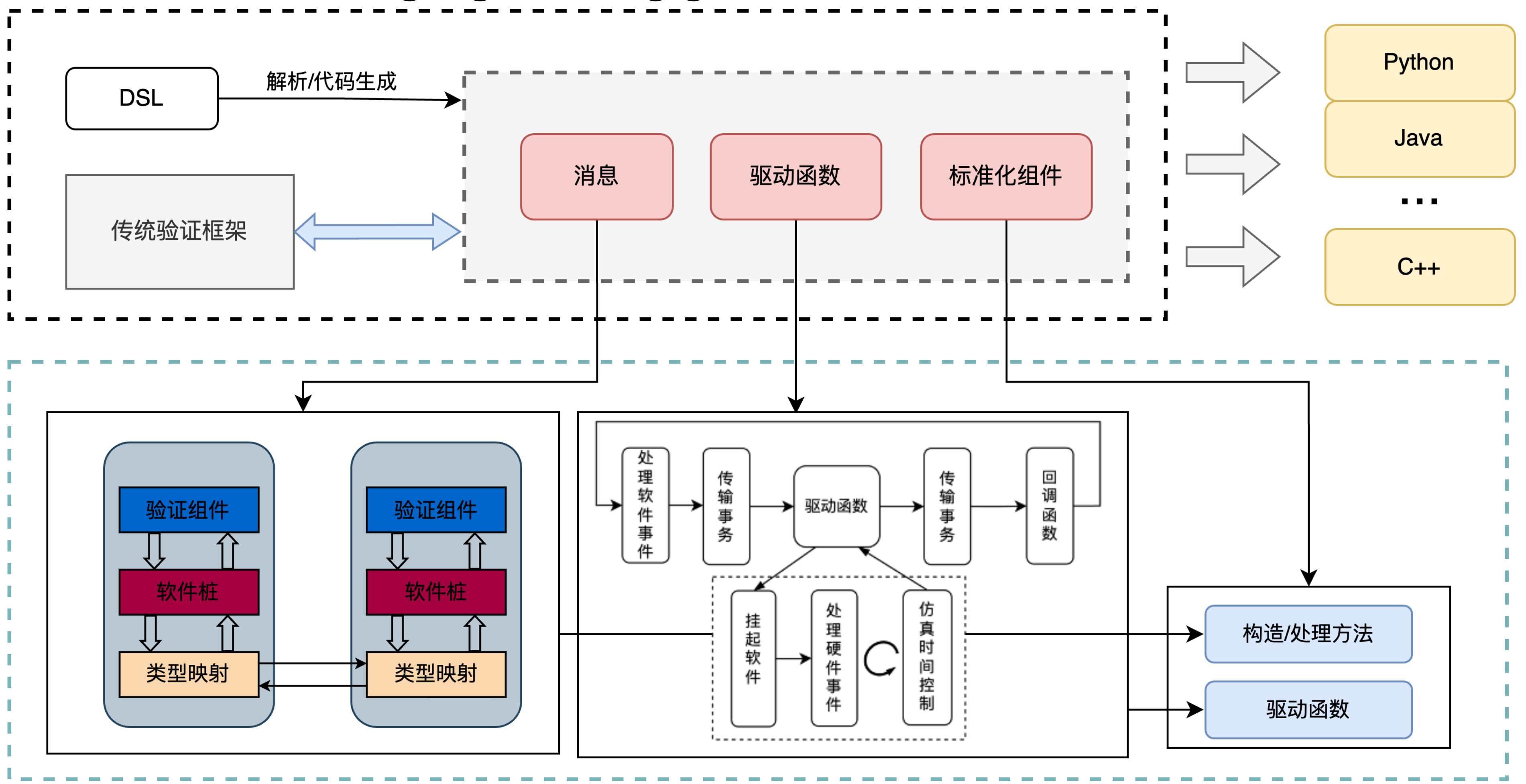




Method Overview

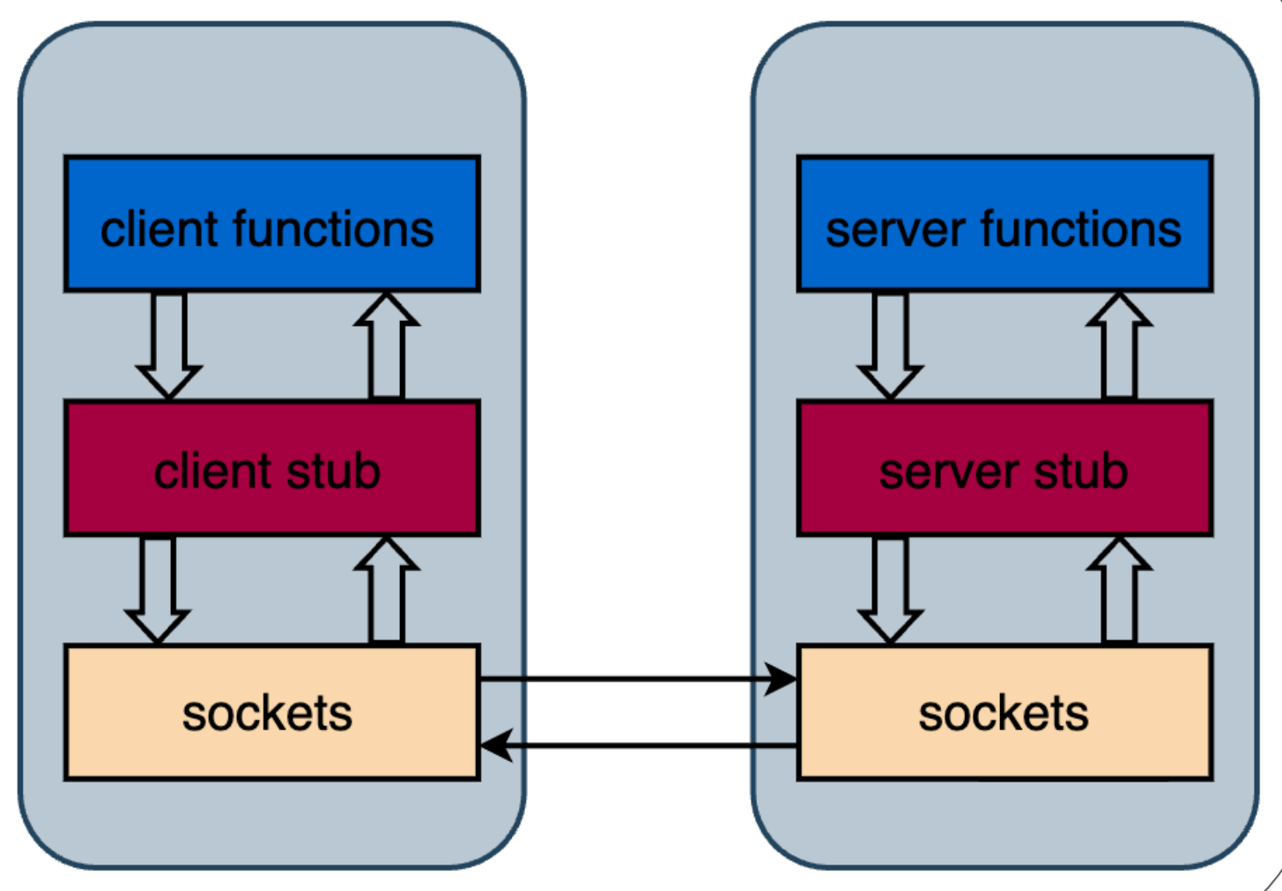
Software-Hardware Co-Verification for Traditional Verification Frameworks

1. We propose a hardware-optimized message-passing architecture enabling software-driven hardware verification, systematically encapsulated into a unified framework with automated multi-language binding generation

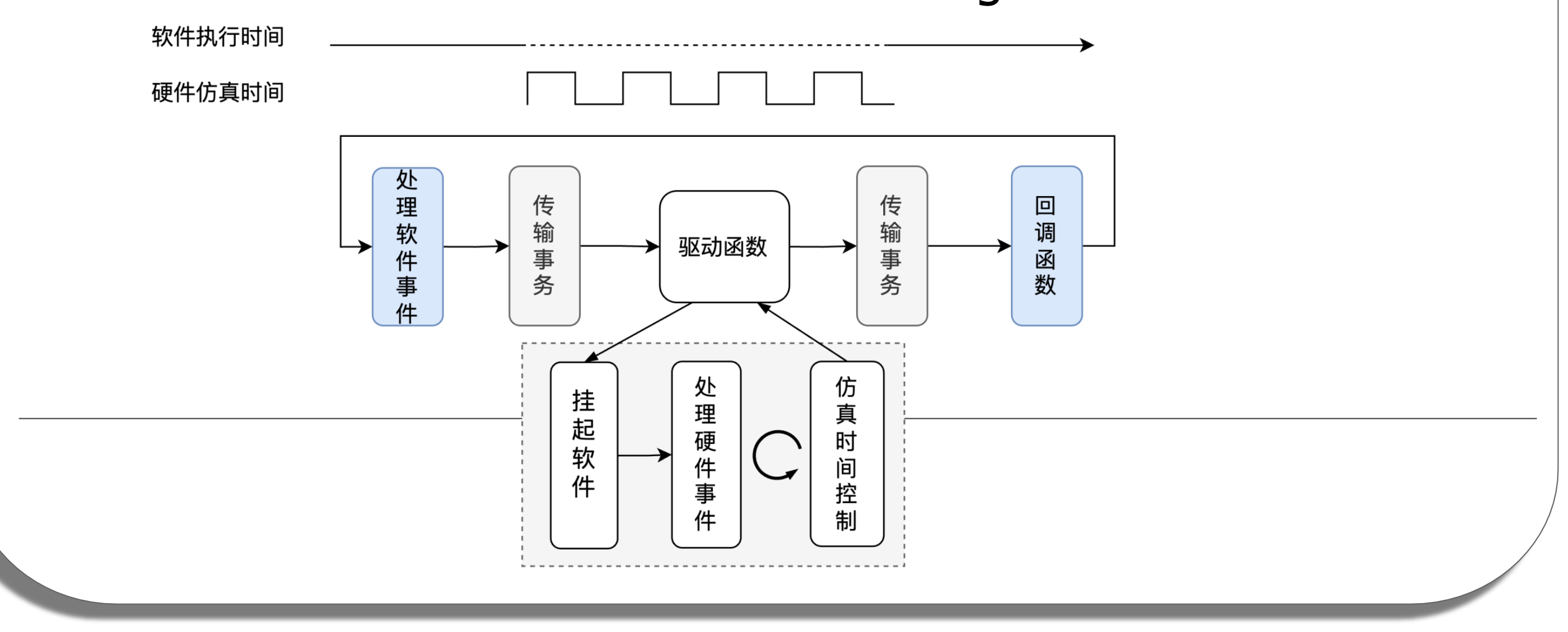


Method

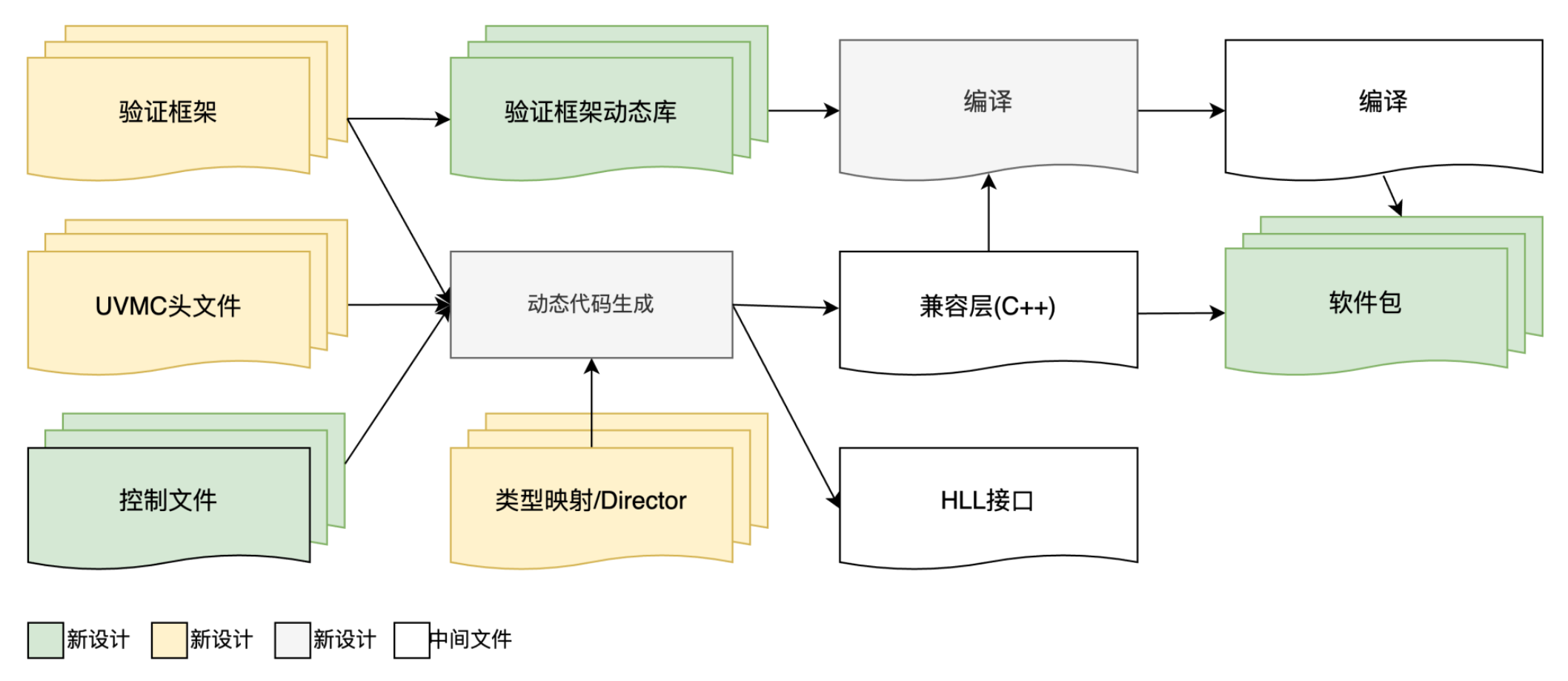
we developed a new Socket type to establish a data path between software tools and hardware frameworks.



One of the key innovations of our work is the decision to drive traditional hardware through software



We compiling hardware designs into dynamic libraries and encapsulating them as software packages.



Future work

Traditional UVM-based verification requires specialized expertise in SystemVerilog, creating a barrier for software engineers who could otherwise contribute valuable test scenarios. By encapsulating UVM within a software-accessible package, our approach allows: Software teams to develop and contribute test scenarios without in-depth UVM knowledge. AI-driven verification where machine learning models can generate and optimize test cases dynamically.

Cross-domain collaboration, enabling both hardware and software engineers to work within a unified verification environment.

.