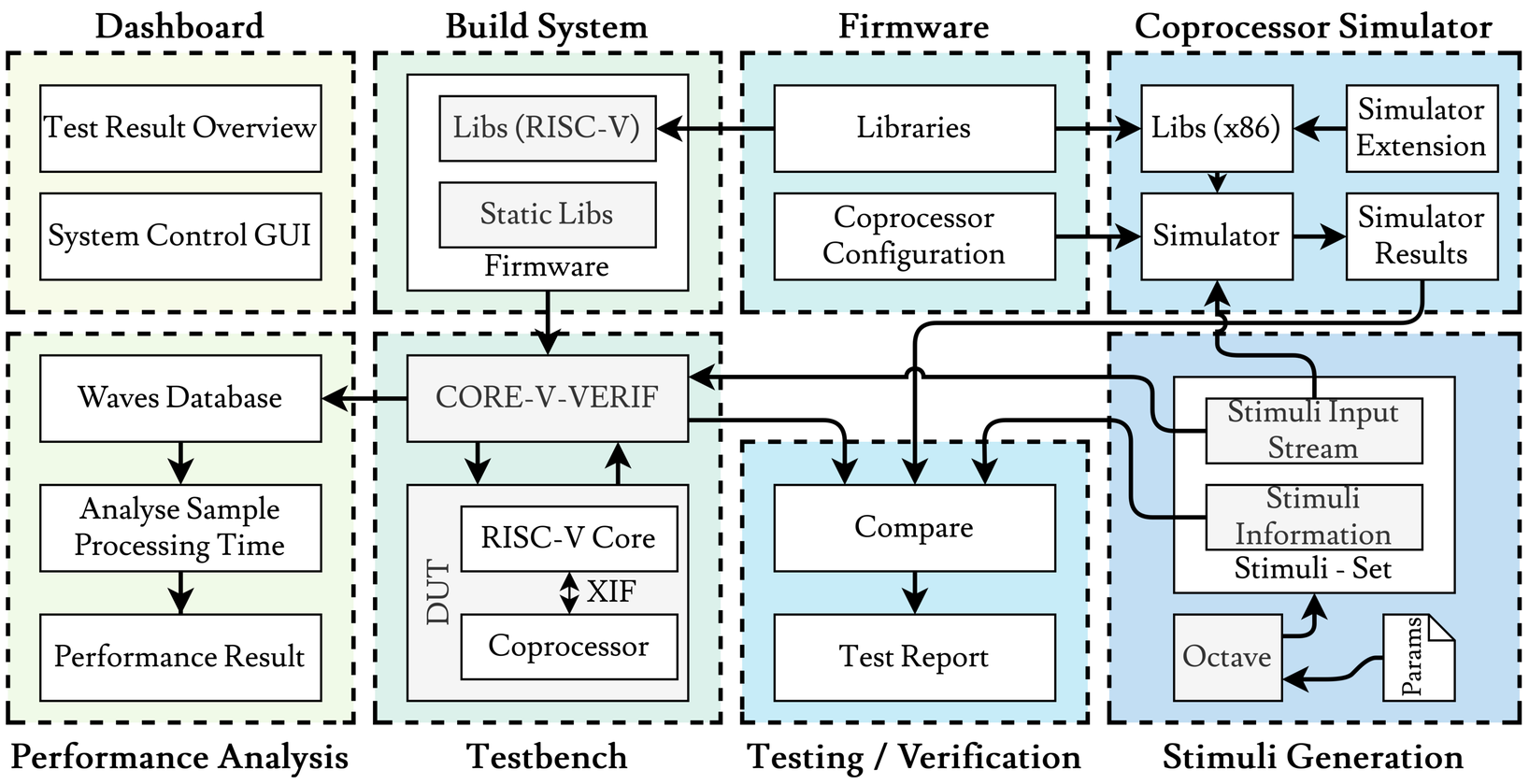
Enhancing the RISC-V Firmware Development Workflow through a Flexible Tooling Environment

Christian Sommerauer[1], Christian Seifert[1], Christian Steger[1] and Tiberio Fanti[2]
[1]Institute of Technical Informatics, Technical University of Graz
{christian.sommerauer, christian.seifert, steger}@tugraz.at
[2]NXP Semiconductors Austria GmbH and Co KG
tiberio.fanti@nxp.com

Overview

- Division of the Firmware Development Process
- Eight specialized Submodules
  - Whole development process covered
- No RTL Code for most of the development process required
- Fast Prototyping of
  - ISA Extensions
  - Algorithms
- Easy Adaptation due to modular structure

- PoC on NFC-Transceiver Use Case



Dashboard

- Provides User with a Graphical User Interface
- Selection of Firmware Versions
- Environment Configuration dependent on Development Stage
- Test Result Visualization

Build System

- Automation of firmware building process
- Based on User Configuration
- Additionally compiles and generates files for analyzing the resulting binaries on an assembly level

Firmware

- Repository for:
  - Firmware Versions
  - Required Libraries
  - Configuration Files
- Supplies Build System and Coprocessor Simulator

Coprocessor Simulator

- Used during earlier Development Stages
- Faster Execution Speed
- No RTL-Code required
- Runs on Host-machine
- Testing of new Ideas or ISEs for the coprocessor

Performance Analysis

- Analyzes Sample Processing Time
- Uses Output from the Testbench
- Used for Data Gathering and Visualization of Results

Testbench

- Runs the Core-V-Verif Project [2] provided by the OpenHW Group [1]
- Simulates Firmware running on the selected core [3]
- Core coupled with a Coprocessor (RTL-Code)

Testing / Verification

- Compares the results from the Testbench or Coprocessor Simulator with Stimuli Information
- Generates a Test Report for the Dashboard

Stimuli Generation

- Provides the test environment with a stream of samples as input
- Octave-based framework
- Use-Case specific params and resulting output stream

Results / Conclusion

- **Introduction of modular Tooling Environment for Firmware Development**
  - **Use Case Flexible**
  - **Easily Extendable or Customized as needed**
- **Proof of Concept implementation showed an increase in productivity**
- **Approach may be of interest for teaching environments**

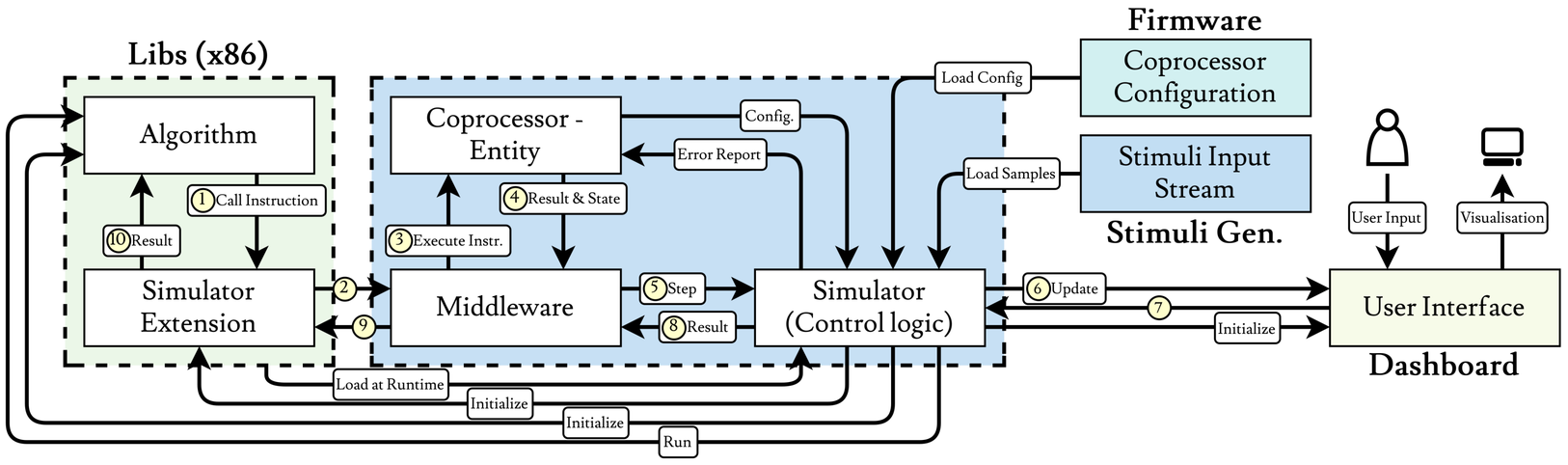


Figure. Coprocessor Simulator Overview as implemented in PoC

TRISTAN Project

References

[1] OpenHW Group. Accessed: 2024-03-18. 2024. url: https://www.openhwgroup.org/.
[2] OpenHW Group. CORE-V Verification Testbench. Accessed: 2024-03-18. 2024. url: https://github.com/openhwgroup/core-v-verif.
[3] OpenHW Group. CV32E40X RISC-V Core. Accessed: 2024-03-18. 2024. url: https://github.com/openhwgroup/cv32e40x.

Extended Abstract

TRISTAN Project has received funding from the Chips Joint Undertaking (Chips-JU) under the grant agreement nr. 101095947. Chips-JU receives support from the European Union's Horizon Europe's research and innovation programme and Austria, Belgium, Bulgaria, Croatia, Cyprus, Czechia, Germany, Denmark, Estonia, Greece, Spain, Finland, France, Hungary, Ireland, Israel, Iceland, Italy, Lithuania, Luxembourg, Latvia, Malta, Netherlands, Norway, Poland, Portugal, Romania, Sweden, Slovenia, Slovakia and Turkey.