ENABLING RISC-V CI IN OPEN-SOURCE PROJECTS: CHALLENGES AND SOLUTIONS

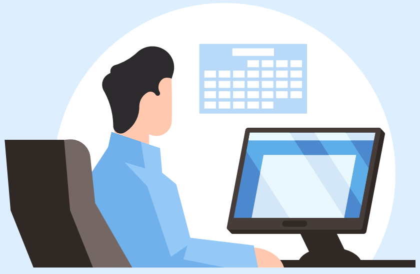
The adoption of RISC-V as a viable architecture for open-source software development is gaining traction. However, a major challenge remains: ensuring continuous integration (CI) support for RISC-V in upstream projects. We faced this issue at Samsung and addressed it by enabling RISC-V CI for several Freedesktop.org (FDO) projects, including Pixman and GStreamer Orc, and we are currently extending the support to the Opus codec. This work presents our approach to enabling RISC-V CI in FDO projects, addressing the challenges of testing architecture-specific optimizations without native hardware support. We detail our implementation of Docker-based GitLab runners with QEMU emulation, enabling automated multi-architecture testing while minimizing infrastructure overhead. Our work not only enhances software quality by enabling automated testing for RISC-V, but it also provides a framework for future contributions to seamlessly integrate RISC-V into open-source CI ecosystem.

DEVELOPER STORY

Steve is a developer working on implementing RVV optimizations in an open source library.

He prepares a patchset, tests it on his RISC-V board, and submits the changes upstream.

However, the maintainers reject his proposal as they have no way to test the new architecture.



SOLUTION AND GOALS

Generic, multi-architecture, Continuous Integration templates.

- Provide CI coverage for **all supported architectures**.
- Use **native runners** where applicable.
- Build and test with both **GNU and LLVM** toolchains.
- Execute tests for **all supported SIMD backends**.
- Generate a consolidated **coverage report**.

PIPELINE STRUCTURE

We developed reusable GitLab CI templates to build **multi-architecture Docker images**, leveraging extensive cross-architecture support of Debian.

To enable comprehensive testing, our images included necessary **toolchains** (GNU, LLVM), required **library dependencies**, and tooling for **analyzing test results**.

Using a **multi-stage**, **composable** Docker image approach, we ensured flexibility, allowing projects to use pre-built images or extend them for project needs.

Whenever possible, we prioritized **native execution** to simplify dependency management and enable coverage analysis. However, for certain architectures, **cross-compilation** was required, in which case we focused solely on correctness verification.

Initially, Pixman's CI workflow ran tests only with the GNU toolchain, leading to **regressions** and **overlooked issues** in LLVM builds.

Architecture-specific code (e.g., with compiler intrinsics) can behave differently between compilers, necessitating dual-toolchain testing. Including **both GNU and LLVM** in our workflow uncovered several **previously unreported bugs**, improving the library's overall reliability.

The CI workflow executes tests across **all supported architecture-specific backends**. For instance, x86 builds reuse binaries across MMX, SSE2, and SSSE3 SIMD implementations allowing for running the tests in parallel.

Similarly, for RVV, we tested **multiple vector register lengths (VLENs)** to identify configuration-specific issues. Since the development hardware we used supported the VLEN of 256, automated testing of other configurations was essential to ensure **early discovery of potential issues for future RISC-V targets**.

Native execution simplified **coverage analysis**, but each test run generated separate reports.

Manually reviewing **over 40 reports** was impractical, so we used gcovr tool to merge them into **a unified summary**, producing both human-readable and machine-consumable reports for GitLab's CI system, allowing for seamless integration with GitLab GUI.

BENEFITS FOR DEVELOPERS

- Easily test new revisions on multiple virtual targets simultaneously.
- Reduce the risk of introducing regressions for other architectures when refactoring.
- Boost confidence when submitting code upstream.
- Increase the chances of merging changes for new architecture upstream.

BENEFITS FOR PROJECTS

- Centralize CI maintenance shared across multiple projects.
- Easily integrate the CI pipeline in a new project by following configuration from other projects.
- Reduce time required from maintainers during code review.
- Minimize risk of accidentally introducing regressions into the codebase.

CONCLUSIONS

By enabling RISC-V CI in upstream open-source projects, we have addressed one of the key barriers to RISC-V adoption. Our approach is generic enough that it can be reused in other Freedesktop.org repositories and in external GitLab instances used by other projects.

This work ensures that maintainers can validate RISC-V contributions without requiring dedicated hardware, making it easier for developers to contribute RISC-V optimizations. This success story demonstrates how strategic CI integration can accelerate the growth of RISC-V within the open-source ecosystem.

MAREK PIKUŁA
Samsung R&D Institute Poland
m.pikula@partner.samsung.com

Embedded developer by day, DevOps engineer by night. Marek creates high-quality, well-tested and documented solutions in established technologies while actively exploring the new and shiny. He feels the best in complex projects requiring system-level and in-detail perspectives, connecting multiple domains from hardware through gateware and firmware up to the software running in the cloud.