Evaluating SYCL Support on RISC-V Multicore Architectures: A First Approach

Enrique de la Calle Montilla, Carlos García Sánchez

Fac. Informática, Universidad Complutense de Madrid,
28040 Madrid (Spain)

{encalle,garsanca}@ucm.es

Abstract

- *The increasing adoption of RISC-V architectures calls for the development of a robust parallel programming infrastructure.*
- *The feasibility of SYCL on RISC-V multicore systems remains largely unexplored.*
- *This ongoing study evaluates the compatibility and performance of multiple open-source SYCL implementations on RISC-V multicore architectures.*
- *Preliminary performance and compatibility testing using the SYCL-Bench suite showed that most benchmarks have been successfully executed on RISC-V multicore architectures.*

1. Experimental Environment

1.1 Banana Pi BPI-F3 board

The Banana Pi BPI-F3 is a RISC-V development board designed for AI and embedded applications. The board features the SpacemiT K1 System-on-Chip (SoC), equipped with an octa-core RISC-V processor (in-order) with 256-bit RVV 1.0 vector support. Additionally, it also incorporates a NPU and an GPU with support OpenCL.

1.2 SYCL

SYCL supports RISC-V architectures as its main implementations rely on Clang, which has included official RISC-V support since version 9.0. However, full compatibility requires both SYCL and its target backends to support RISC-V. While Intel's OpenCL runtime is limited to x86 and x86-64, Portable OpenCL (PoCL)[1] offers an alternative, providing OpenCL support for RISC-V through a pthreads-based implementation. Additionally, OpenMP offloading, as facilitated by AdaptiveCpp, can serve as a viable target backend for SYCL on RISC-V.

1.3 Benchmarks

To evaluate SYCL support on RISC-V-based architectures, the suite SYCL-Bench [1] has been selected. Two of the three available categories have been evaluated: (1) microbenchmarks assessing memory bandwidth, arithmetic throughput, and scheduling latency; (2) application benchmarks spanning linear algebra, image processing, and scientific simulations.

Some code adaptations on SYCL-Bench should be performed to ensure RISC-V compatibility. These changes, along with compilation scripts for result replication, are available in our public repository[2].

1.4 Setup

The following software versions were used for benchmark execution:

- **Host compiler:** Bianbu Clang 18.1.8 (11bb4)
- **AdaptiveCpp:** Commit `c0a69b2`
- **DPC++:** Commit `903279c`
- **PoCL:** Release 6.0, Commit `952bc55`
- **SYCL-Bench:** Fork based on Commit `aabfb41`

2. Discussion

Figure 1 presents microbenchmark results for AdaptiveCpp with OpenMP (Acpp + OMP), AdaptiveCpp with Portable OpenCL (Acpp + PoCL), and Intel DPC++ with Portable OpenCL (DPC++ + PoCL). Notable performance disparities at lower-level operations across implementations and backends are evident, particularly the bandwidth slowdown in Acpp + PoCL, consistent with prior observations [1].

More consistent results across implementations are observed in the application benchmarks (Figure 2), with the most notable discrepancies appearing in the *syrk* and *mvt* benchmarks.

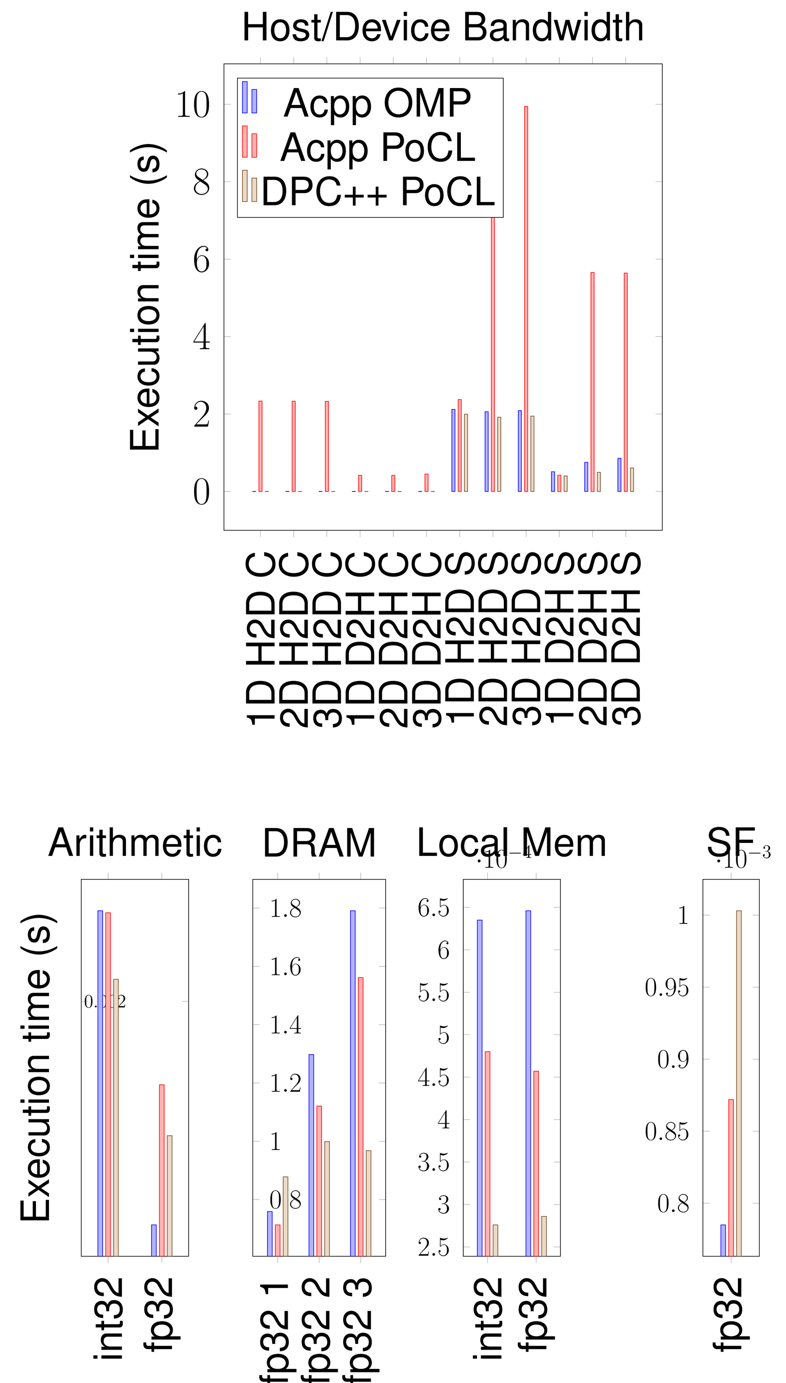

Figure 1: *Microbenchmarks*

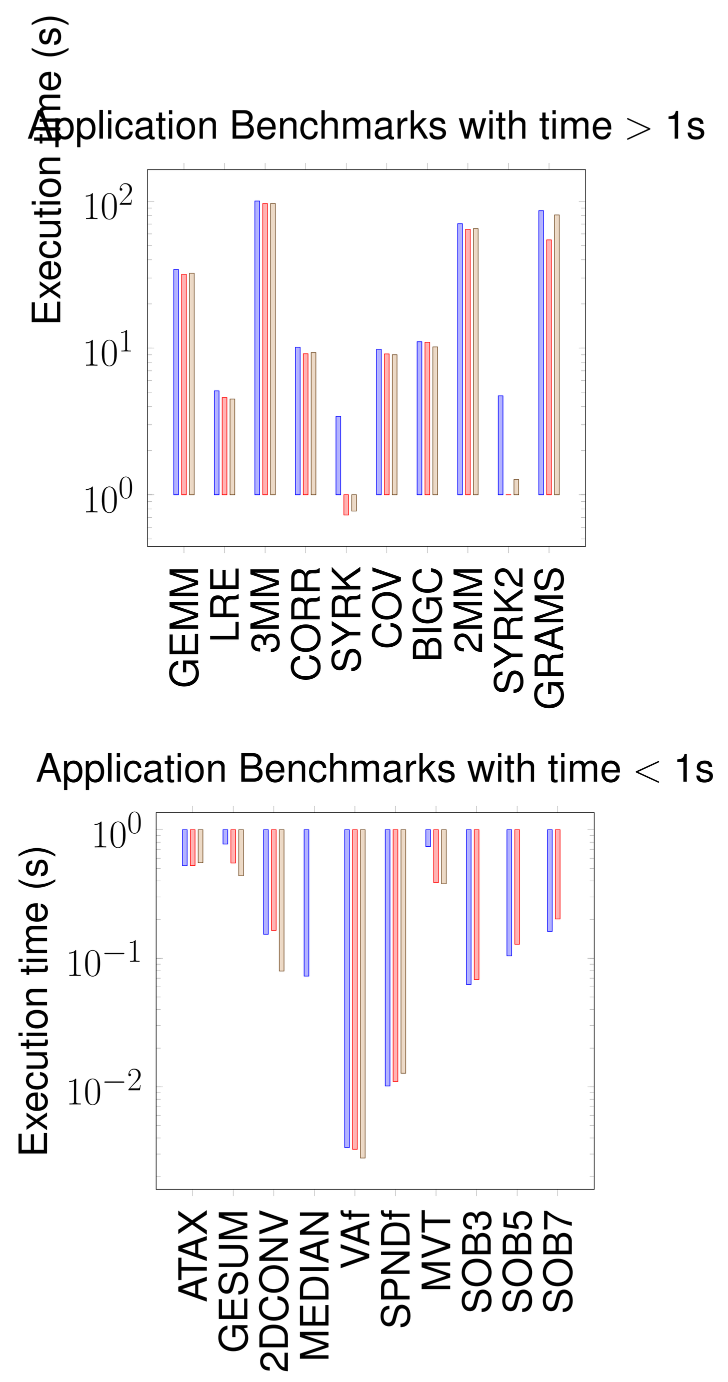

Figure 2: *Application Benchmarks*

2.1 Execution errors

Some of the benchmarks couldn't execute properly, being Acpp+PoCL the worst offender. Such benchmarks are listed below: Hierarchical executions and atomics float operations failed on Acpp + PoCL, *Median* failed on PoCL and *Sobel* failed on DPC++

2.2 Verification errors

Two benchmarks exhibited verification issues across all implementation-backend combinations: *Molecular Dynamics* and *LinearRegressionCoeff*. Notably, no benchmark failed verification in one implementation while passing in another.

3. Conclusions

Most of SYCL benchmark suite for parallel performance evaluation was executed on a BPI-F3 board equipped with a RISC-V octa-core processor. The results revealed discrepancies across different implementations, with certain benchmarks failing to execute on specific backends. However, only a small subset of tests failed consistently across all available backends, demonstrating the viability of executing portable parallel code on RISC-V architectures.

As future work, further exploration is needed to enhance performance portability and optimize vectorization techniques on RISC-V-based systems. Specifically, AdaptiveCpp facilitates automatic vectorization in the OpenMP backend through an LLVM pass, and additional research will focus on integrating and evaluating the RISC-V Vector Extension (RVV) within SYCL. Moreover, a more in-depth analysis of execution failures and verification discrepancies will be conducted, alongside comprehensive testing of the runtime suite. Besides, Intel provides an experimental open-source OpenCL/Vulkan runtime for RISC-V architectures via the OneAPI Construction Kit, making it a promising platform for further analysis.

References

[1] Crisci, L., Carpentieri, L., Thoman, P., et al.: Sycl-bench 2020: Benchmarking sycl 2020 on amd, intel, and nvidia gpus. In: Proceedings of the 12th International Workshop on OpenCL and SYCL. pp. 1–12 (2024)

[1] PoCL: `https://github.com/pocl/pocl`
[2] `https://github.com/101001000/sycl-bench`

Supported by the EU (FEDER), the Spanish MINECO under grants PID2021-126576NB-I00 funded by MCIN/AEI/10.13039/501100011033.