Experiences of running an HPC RISC-V testbed

Providing a complete ecosystem where users can experiment with RISC-V for their HPC and ML codes

Deceptively simple initial steps

Common HPC libraries and utilities are prebuilt for RISC-V, and it is possible to network different types of RISC-V device and for these to work (fairly) seamlessly together with MPI

netCDF FFTW PETSc slurm workload manager

Our testbed feels like any other HPC machine, with libraries available as modules and RISC-V compute managed via Slurm

Challenges that we faced

However, once you scratch the surface then several limitations start to become apparent

Software tooling: There is a lack of profilers, and pre-built libraries tend to be scalar only, it is crucial we address these to support HPC workloads

OS kernels: Provided Linux kernel can be overly restrictive, e.g. can not compile new modules directly into the DongshanNezhaSTU's kernel due to the protected bootloader

Hardware availability: Difficult to get hold of physical RISC-V hardware, especially in a manner compatible with procurement!

Full details at https://arxiv.org/pdf/2305.00512.pdf

Funded by ExCALIBUR H&ES

ExCALIBUR H&ES

https://excalibur.ac.uk/

The ExCALIBUR program addresses the challenges and opportunities offered by computing at the exascale

The Hardware and Enabling Software (H&ES) component aims to provide novel hardware in the form of testbeds. Our testbed enables HPC developers to experiment with RISC-V

30 second summary

- High Performance Computing (HPC) is the use of supercomputers to simulate traditional scientific & engineering workloads, as well as machine learning. Entering the era of exascale supercomputing, an important question is the future HPC hardware
- RISC-V is a very interesting potential technology to power next-generation supercomputers, but we need to enable HPC developers and users to experiment with the technology in order for it to grow in popularity and be accepted
- In setting up a RISC-V HPC testbed, we have learnt lots around the successes and limitations of the RISC-V ecosystem and developed several tools and insights to support the wider use of RISC-V for HPC workloads

Physical RISC-V CPUs

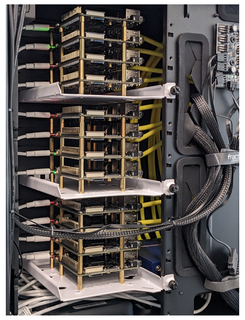
Physical RISC-V CPUs, which represent a variety of different technologies and capabilities, are provided as compute nodes. Compilation is undertaken on the login node which also runs Slurm and has common HPC libraries and compilers available

The testbed currently contains the following type of nodes:

- HiFive Unmatched (quad core U740)
- StarFive VisionFive V2 (quad core U74) and V1 (dual core U74)
- Allwinner D1-H (C906 CPU)
- Lichee RV Dock (C906 CPU)
- MangoPi MQ-Pro (C906 CPU)

These cores contain the 0.7 version of the new vectorisation ISA specification, enabling experimentation with SIMD.

- And more types of RISC-V node added as they become available!

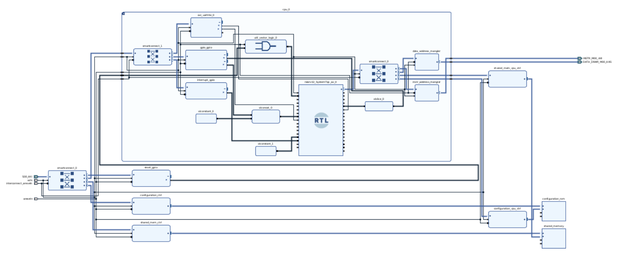


Physical boards enable easy access to RISC-V, however RISC-V is moving very quickly and-so can be somewhat behind the cutting-edge state of the art

Access to cutting edge soft core RISC-V designs

- We provide numerous pre-built soft cores in a catalogue which can be loaded by users. Additional configurations of these can be provided as required
- All soft cores run on a state of the art AlphaData P101 which provides the Versal FPGA
- We are currently integrating RISC-V based accelerators for HPC developers to explore

For example, the image on the right illustrates a single-core NeoRV32 (the central large block) with other blocks providing infrastructure support. This enables the CPU core to access memory, GPIO, UART and interact with the host machine



Free access

Users from across the world welcome

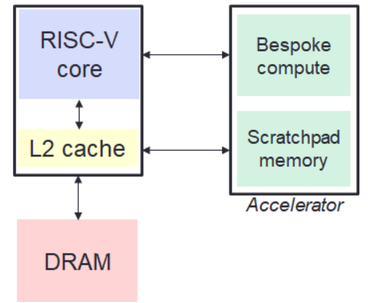
riscv.epcc.ed.ac.uk

Corresponding Author:
Nick Brown, EPCC
n.brown@epcc.ed.ac.uk

Why RISC-V for HPC?

HPC is currently rather commoditised, using off-the-shelf CPUs and GPUs

- RISC-V enables us to leverage bespoke CPUs optimised for HPC workloads
- RISC-V accelerators for common workloads (e.g. linear algebra, FFTs etc) or number representation (e.g. Posits)
- Ubiquity via stealth, RISC-V embedded in other technologies such as networking

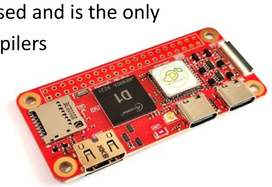


Case study: v1.0 vectorisation on v0.7 hardware

RISC-V®

- **Problem:** Current physical RISC-V cores, such as C906, tend to support version 0.7 of vectorisation, whereas version 1.0 has been released and is the only version supported by up-to-date/current/upstream compilers
- **Solution:** We have developed a tool that manipulates the generated assembly code, to backport executables so that they comply with 0.7 vectorisation standard

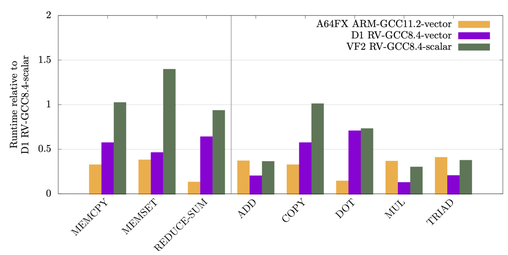
https://github.com/RISCVtestbed/rvv-rollback



Full details at https://arxiv.org/pdf/2304.10324.pdf

Exploratory benchmarking

We are undertaking benchmarking to understand the relative performance of the RISC-V cores and aspects such as vectorisation



Here exploring several RAJAPerf kernels across technologies. RISC-V D1 (vector) and VF2 (scalar) normalised against D1 scalar, A64FX (vector) normalised against A64FX scalar

Full details at https://arxiv.org/pdf/2304.10319.pdf