Work-In-Progress: Accelerating Numpy With OpenBLAS For Open-Source RISC-V Chips

Cyril Koenig[1], Enrico Zelioli[1], Frank K. Gürkaynak[1], Luca Benini[1,2]
[1]Integrated Systems Laboratory, ETH Zurich
[2]Department of Electrical, Electronic, and Information Engineering, University of Bologna

1. Motivation

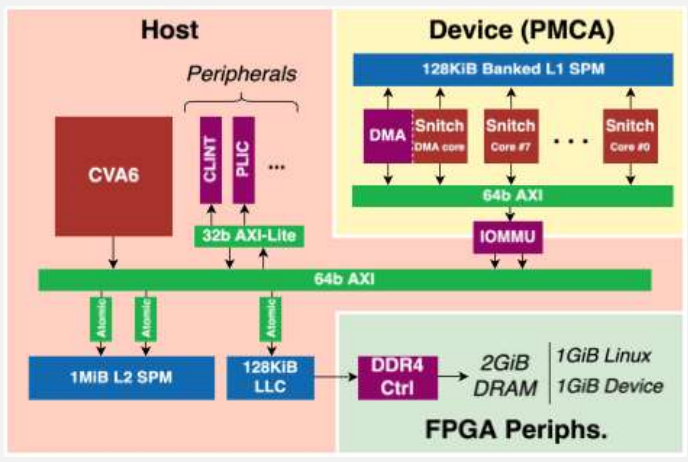
RISC-V allows for building general-purpose computing platforms with programmable accelerators around a single open-source ISA. However, leveraging heterogeneous SoCs within high-level applications is a tedious task. For this purpose, previous works [1] proposed dedicated heterogeneous BLAS implementation. In this work we propose:

- ***An extension of OpenBLAS with heterogeneous kernels support using OpenMP offloading.***
- ***A demonstration of an accelerated Python application using a RISC-V programmable accelerator.***

2. Platform Architecture

The proposed platform contains:

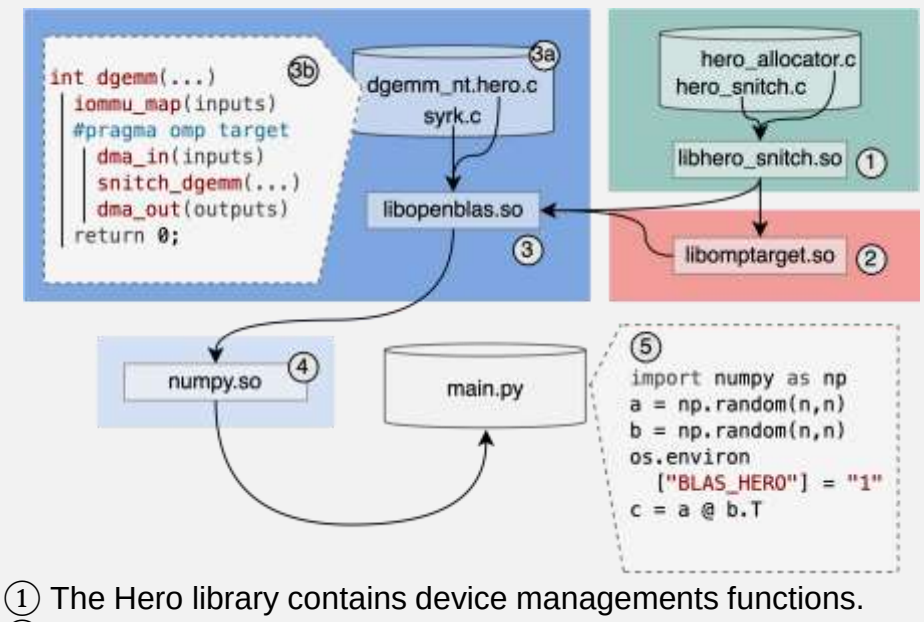
- **Linux** capable CVA6 **core** *(rv64gc)*
- **Programmable Many-Core Accelerator** *(rv32imafd)*
- **RISC-V IOMMU** [2] with four IO-TLB entries



source code and FPGA flow

paper

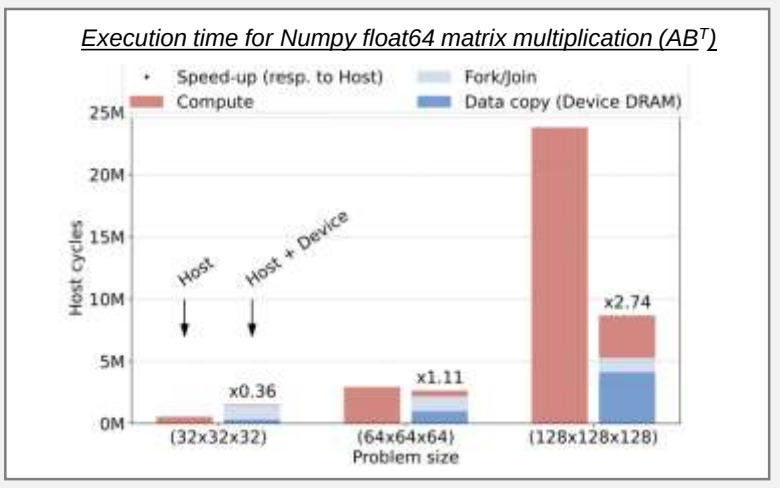
3. Software Architecture



① The Hero library contains device managements functions. ② The OpenMP target library contains the callbacks for the OpenMP API. ③ The OpenBLAS library contains computing kernels for host and/or device. ④The Numpy package is linked against OpenBLAS. ⑤ The user application imports Numpy.

4. Performance Evaluation

We evaluate the platform and OpenBLAS implementation using a Python example application in ⑤ and measure the execution time using the `os.time()` function.



We show that for matrices of 128 x 128 double precision elements, the accelerated implementation is **2.74 times faster than the OpenBLAS implementation** optimized for RISC-V.

The heterogeneous execution is split in the following regions:

- **Fork/Join** for waking the device with OpenMP offloading
- **Data copy** to copy inputs/outputs in device buffers
- **Compute** to compute the output in device buffers

The **data copy is still significant**: 47% of the total runtime. However, future work will leverage the **IOMMU** that **can significantly reduce this overhead**. From previous work [3] on the same platform, we expect IO mapping to be 7.5x faster than data copy for this problem size.

OpenMP target allows for easily porting this implementation to new platforms (for instance *rv32imafdv* accelerators).

5. Conclusion

- The **platform RTL** is **available on Github** with its FPGA flow.
- Using **OpenBLAS leverages** optimized RISC-V kernels for **the host**.
- It is possible to **extend OpenBLAS** with OpenMP offloading to use **programmable RISC-V accelerators**.
- In our preliminary result, the heterogeneous RISC-V platform shows **a 2.74x speedup in Numpy matrix multiplication**.

References

[1] N. Rao Miniskar, et Al. “RIS-BLAS: Towards a Performance Portable and Heterogeneous BLAS Library”
[2] M. Rodríguez, et Al. “Open-source RISC-V Input/Output Memory Management Unit (IOMMU) IP”
[3] C. Koenig et Al. “Evaluating IOMMU-Based Shared Virtual Addressing for RISC-V Embedded Heterogeneous SoCs”