Using the TUM Uncore Environment for RISC-V for Teaching, AI and Quantum Computing

Martin Schulz
Technical University of Munich
schulzm@in.tum.de

Alexis Engelke
Technical University of Munich
alexis.engelke@tum.de

Carsten Trinitis
Technical University of Munich
trinitic@in.tum.de

1 Problem Statement

RISC-V is not only a good match for innovative research, from AI to Quantum Computing, but also for teaching and education. Both research and teaching with RISC-V are frequently conducted on FPGAs, as they offer a good trade-off between easy-to-implement simulations and high performance. However, the development of processor cores on FPGAs comes with a rather high entry barrier: the required uncore logic to connect, feed, test and use a newly designed core is non-trivial and requires significant work. Further, each core design project needs to communicate with the FPGA host to enable easy debugging and performance analysis. As a consequence, a significant amount of engineering overhead must be invested, detracting from the actual research or education targets.

2 The TUM Uncore Environment for RISC-V

To address these issues, we have developed a small and simple, yet flexible and easy to use *uncore* environment to simplify the VHDL-based development of multi-core RISC-V processors from scratch on FPGAs [1]. It provides a cache hierarchy to support multiple cores, atomic memory operations, and a generic interface for accessing memory and I/O components. To account for comparably low memory latency resulting from the lower clock rate of the FPGA itself, the access latency to the caches and the main memory is configurable. We use a bus-based communication protocol between the three layers, allowing us to replace each layer independently of the others. This design facilitates the development of new processor cores and the evaluation of different memory hierarchies with multi-core support. Furthermore, we can migrate the system quickly to other hardware platforms or emulate it on standard CPUs. All components are implemented in VHDL and new projects include the files to make use of our uncore environment.

3 Use Cases

In the following we describe three different use cases in which we rely on the TUM uncore environment to implement and experiment with custom designs.

A Core for Architecture Education: As part of a student lab for freshmen computer science students, we leverage the uncore environment to provide the foundation for the development of a RISC-V core. To achieve high concurrency, we target small processor cores, which, in addition to RV32IM, also support most of the specified atomic instructions of RV32A [2]. Combined with coherent caches, this also enables synchronization. To facilitate multi-core programming even further, we also implement several instructions from the privileged instruction set [3] to support, e.g., unique HART IDs and interrupt handlers. To keep the RISC-V cores small enough to fit many of them on a single FPGA, we implement individually scalar cores without further optimizations, like pipelining or out-of-order execution, although given sufficient space on FPGA´s more complex cores could a be added.

BB-KI AI Teaching Lab: Besides classical application fields, artificial intelligence has attracted enormous interest in the research community. While big companies like Xilinx, NVIDIA, ARM, and Intel have started integrating more and more AI features into their platforms,

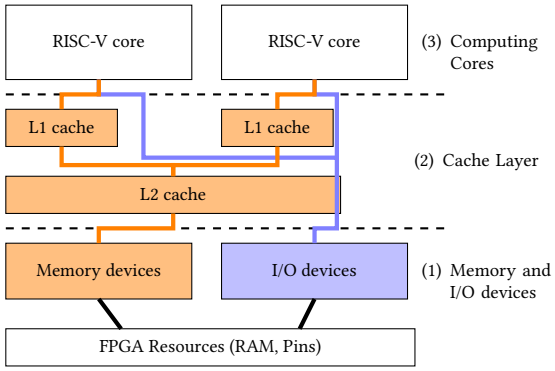

Figure 1. The TUM Uncore Environment

AI hardware is not yet taught at universities as it probably should be. For this reason TUM has started a new program, together with the University of Potsdam, that aims at teaching exactly this: by targeting a multidisciplinary group of students (hardware, AI, and application) the goal is to provide practical teaching and team work in a realistic environment, i.e., real chip production in factories in Germany. Also here, we rely on our uncore environment, as it allows us to quickly prototype AI accelerators and integrate it with RISC-V host cores, as the one described above.

Quantum Control Processors: The final use case described on our poster opens a new paradigm in computing, the use of quantum computing. While the actual processing techniques follows fundamentally different principles, the control system around it is still traditional von Neumann. Current approaches control a set of actuators, often microwave, via simple and coarse-grained Etherned-based protocols. As quantum systems scale, this will no longer be sufficient. We will need to explore parallelism both in instructions/gates and in the system overall. For this we are designing a new Quantum Control architecture. The TUM uncore environment will provide the foundation for such an integration, enabling rapid prototyping. The work is done as part of the Q-DESSI project within the Munich Quantum Valley (MQV).

4 Wrap-Up

The TUM uncore environment significantly eases the development and testing of RISC-V processing elements and makes the topic approachable to students and researchers. It forms the foundation for several activities at TUM, from core design in student labs, the development of new AI platforms to the research on a quantum control processor, and with that will further push RISC-V as a future architecture for a wide range of applications.

References

[1] Michael Jungmair, Tobias Schmidt, Alexis Engelke, Armin Ettenhofer, Felix Krayer, Jonas Lauer, Malte von Ehren, and Martin Schulz. 2021. A Flexible Uncore Infrastructure for RISC-V Core Development. In *Proceedings of the Fifth Workshop on Computer Architecture Research with RISC-V (CARRV 2021).*

[2] RISC-V Foundation. 2019. *The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Document Version 20190608-Base-Ratified.*

[3] RISC-V Foundation. 2019. *The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, DocumentVersion 20190608-Priv-MSU-Ratified.*

BB-KI is funded by the German BMBF under grant agreement 16DHBKI021.
Q-DESSI is funded by the Free State of Bavaria via the Munich Quantum Valley.