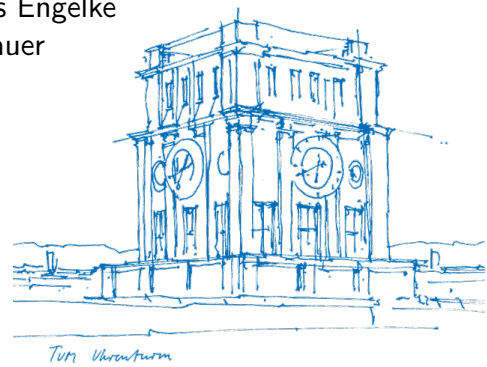
A Flexible Uncore Infrastructure for RISC-V Core Development

Michael Jungmair Tobias Schmidt Alexis Engelke
Armin Ettenhofer Felix Krayer Jonas Lauer
Malte von Ehren Martin Schulz

Chair of Computer Architecture and Parallel Systems
Department of Informatics
Technical University of Munich

CARRV 2021
at ISCA 2021, virtual



Introduction

- RISC-V: rapidly evolving, open, and flexible
- Small base ISA $\rightarrow$ simplified development of fully functional processor cores
- Prototyping on FPGAs:
  - Good tradeoff between implementation effort and performance
  - High entry barrier, time-consuming development of auxiliary components

$\Rightarrow$ Flexible uncore infrastructure:

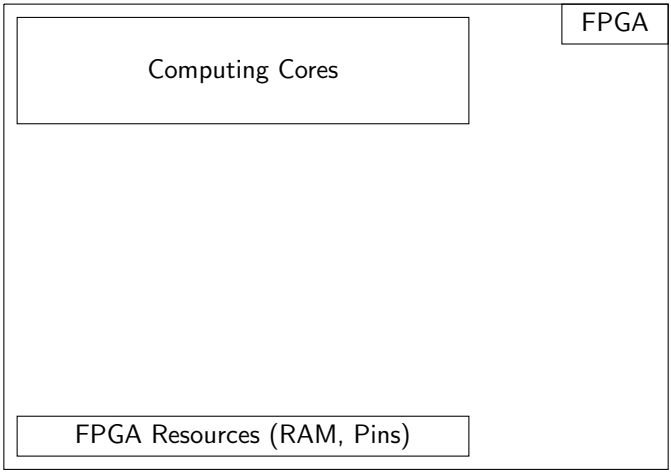
- Lower entry barrier for FPGAs
- Simplify implementation of RISC-V cores in VHDL

Overview

RISC-V Cores on FPGAs

Required components:

- Computing Cores

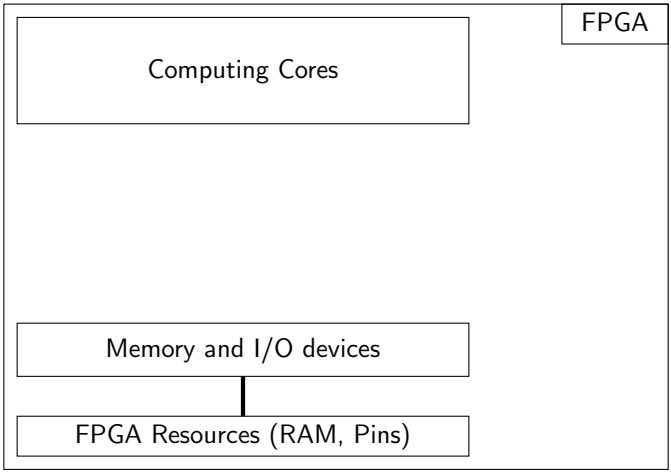


Overview

RISC-V Cores on FPGAs

Required components:

- Computing Cores
- Memory devices
- I/O devices

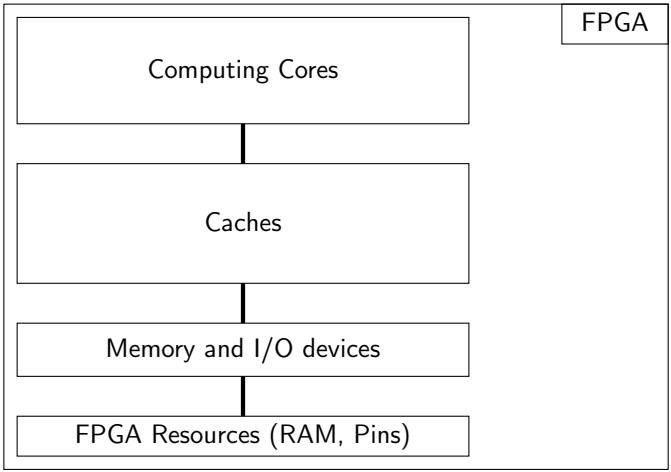


Overview

RISC-V Cores on FPGAs

Required components:

- Computing Cores
- Memory devices
- I/O devices
- Caches



Uncore Infrastructure

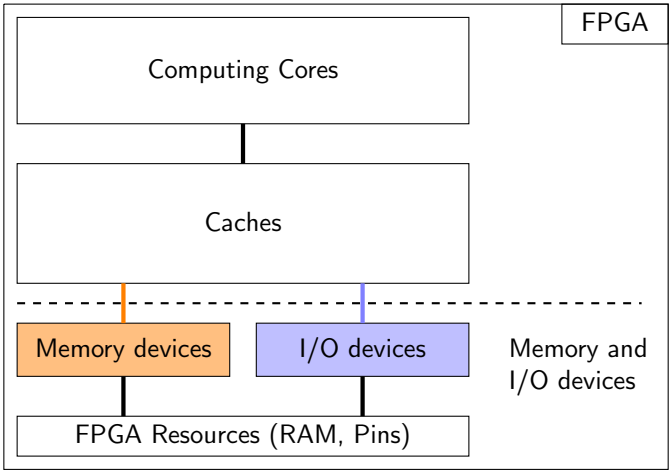
Memory and I/O devices

Storage:

- DDR3 memory
- On-chip memory

I/O (memory-mapped):

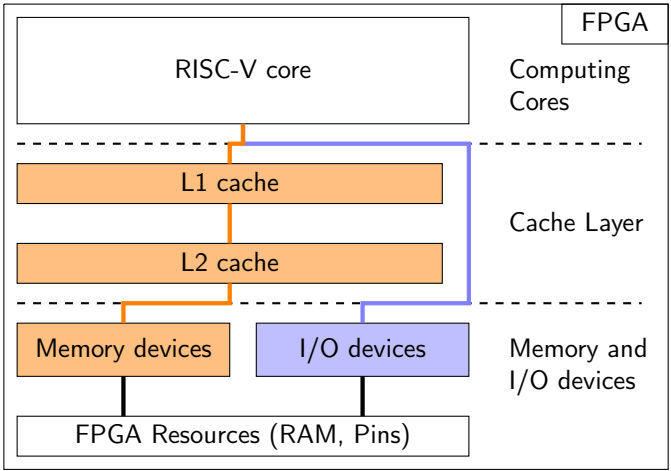
- Hardware clock
- Communication buffers
- ...



Uncore Infrastructure

Caches

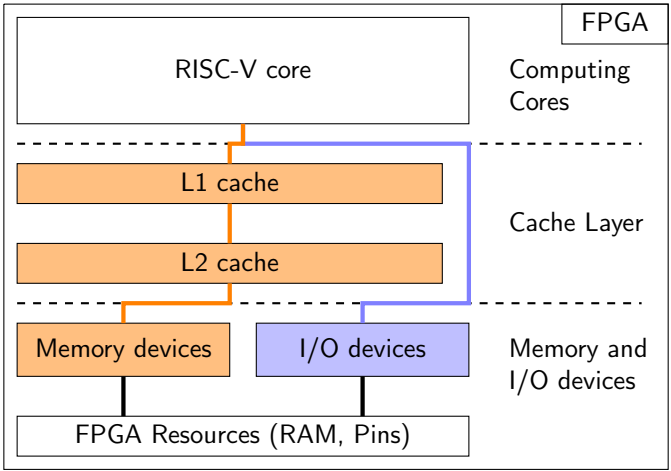
- Write-back cache with write allocation
- Four-way set-associative
- LRU eviction policy
- 512-bit cache lines
- Configurable latencies



Uncore Infrastructure

Accessing the Caches

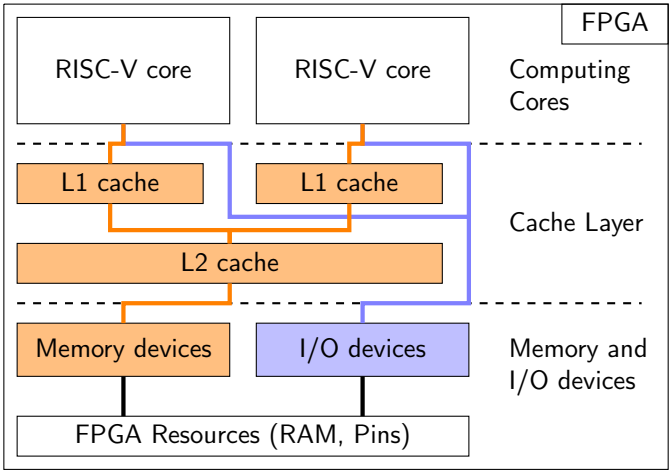
- Most-significant address bit to distinguish memory and I/O devices
- 64-bit aligned write and read operations
- 512-bit reads for instruction buffers



Uncore Infrastructure

Multi-core Support

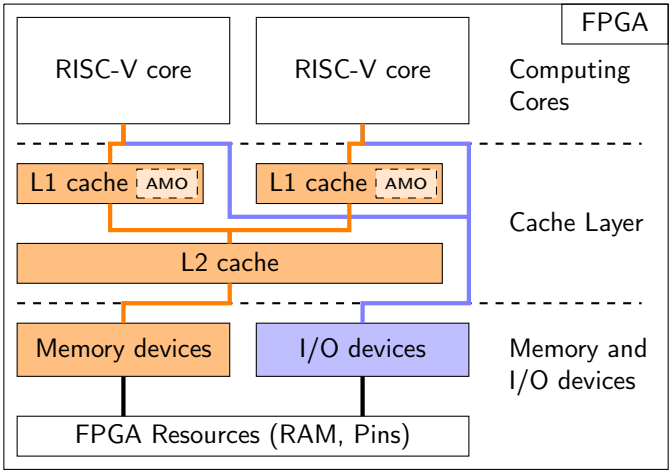
- Duplicate the L1-cache for every core (private cache)
- Shared L2-cache
- MESI cache coherency (directory-based)



Uncore Infrastructure

Multi-core Support

- Duplicate the L1-cache for every core (private cache)
- Shared L2-cache
- MESI cache coherency (directory-based)
- Atomics in L1-cache

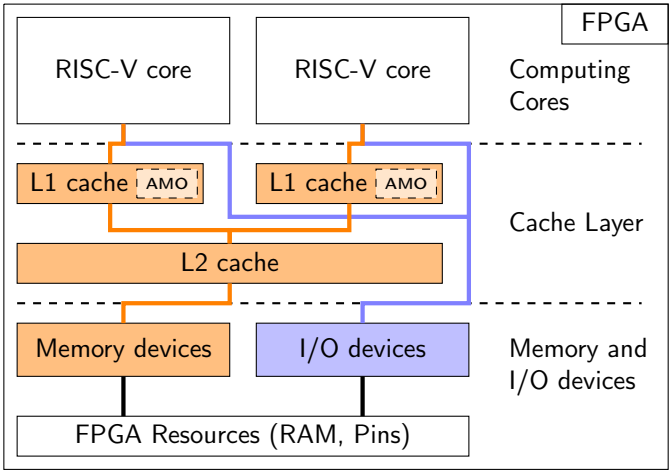


Uncore Infrastructure

Debugging Protocol

Problems:

- Internal state not accessible
- Logic hard to debug
- Booting dynamic programs



Uncore Infrastructure

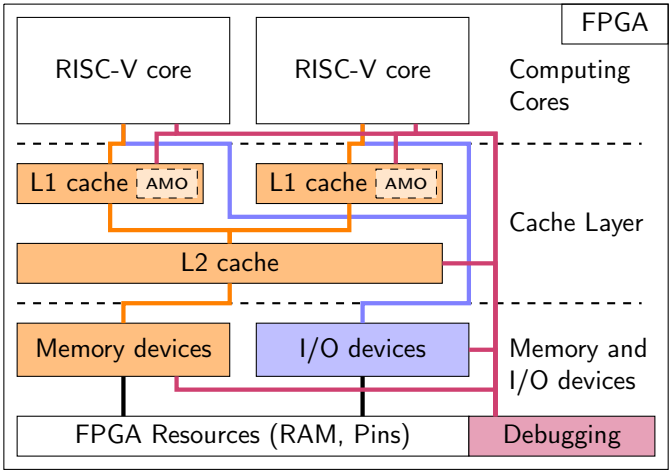
Debugging Protocol

Problems:

- Internal state not accessible
- Logic hard to debug
- Booting dynamic programs

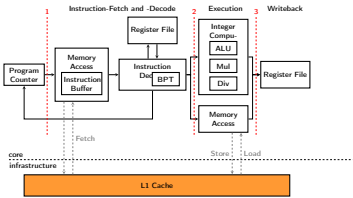
Solution:

- UART-based protocol
- Generic message passing
- Embeddable component

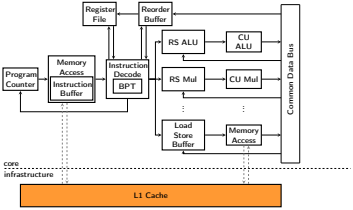


Case Studies

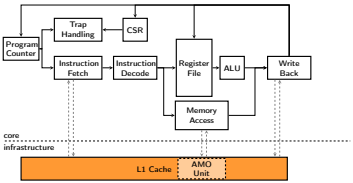
3-stage pipelined core



Out-of-order core



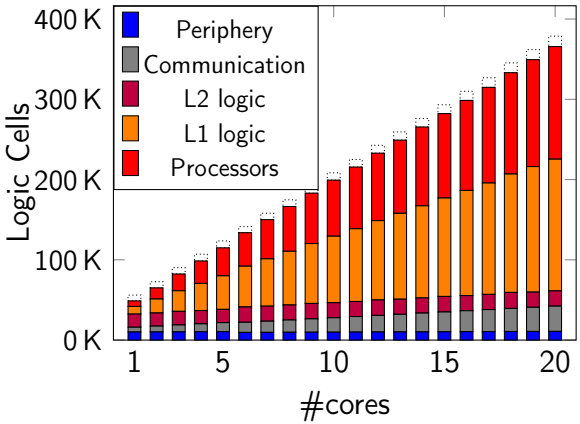
Size-optimized scalar core



Same infrastructure, no changes to the underlying layers.

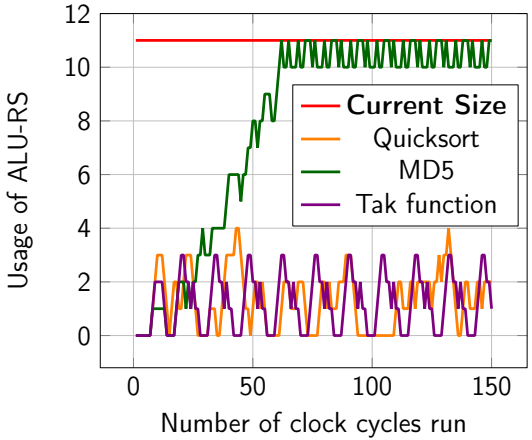
Experiences: Scalability

- Sufficient foundation for developing single- and multi-core RISC-V processors
- Small enough for small boards
- Scales well with an increasing number of cores



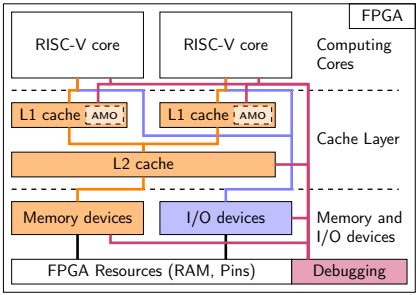
Experiences: Ease of Core Development

- Flexible debugging possible reduce development effort
- Detailed performance statistics for in-depth analysis and further optimizations
- Even freshman undergraduate students were able to implement sophisticated core designs



Summary

- Flexible Uncore Infrastructure for developing VHDL-based RISC-V processors
  - Memory and I/O devices
  - 2-level cache layer (coherent)
  - Atomic memory operations
  - UART-based debugging protocol



- Supports performance-oriented and multi-core designs
- Allows scalable implementation of multi-core systems
- Also sucessfully used for teaching at TUM