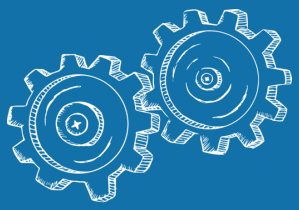

Unified Virtual Addressing for the Vortex Open-Source GPU

Advisor: **Paul Gollob**

Motivation

Modern GPUs support **unified virtual addressing (UVA)**: CPU and GPU share a single virtual address space, so pointers can be passed between host and device without copying or translation. This simplifies programming, but it also means that the GPU must translate addresses and enforce the same access permissions as the CPU. Otherwise, the accelerator becomes a way around process isolation.

Vortex is an open-source, RISC-V based GPU written in SystemVerilog, and a first **virtual memory** implementation for Vortex already exists. This makes it possible to explore UVA and its security implications at the **RTL level**.

In this master project or thesis, you will extend the existing virtual memory support of Vortex towards UVA, e.g., by sharing page tables with the host process, and make sure the GPU respects the host's memory isolation.

Goals and Tasks

- Get familiar with GPU virtual memory and UVA (e.g., in CUDA), and the existing Vortex virtual memory implementation.
- Design a UVA scheme for Vortex, including address-space layout, page table sharing, and permission enforcement.
- Implement the design in the Vortex RTL and adapt the driver/runtime.
- Evaluate correctness, isolation guarantees, and the performance and area overhead in simulation and on FPGA.

Literature

- B. Tine et al.
  Vortex: Open-Source RISC-V GPU
  MICRO 2021
  https://github.com/vortexgpgpu/vortex
- L. E. Olson et al.
  Border Control: Sandboxing Accelerators
  MICRO 2015
  https : / / doi . org / 10 . 1145 / 2830772 . 2830819

Courses & Deliverables

☑ **Master Project**
Project code
Report
Presentation

– OR –

☑ **Master's Thesis**
Initial presentation
Project code
Thesis (60+ pages)
Final presentation

Recommended if you're studying

☑ CS ☑ ICE ☑ SEM

Prerequisites

- Computer architecture, virtual memory, SystemVerilog
- C/C++, ideally OpenCL/CUDA

Advisor Contact

paul.gollob@tugraz.at