TU Wien Informatics

FPGRay

Alexander Reznicek
Masterstudium Technische Informatik

TU Wien Informatics
Institute of Visual Computing and Human-Centered Technology
Research Unit of Computer Graphics
Supervisor: Assoc. Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Assistance: Dipl.-Ing. Hiroyuki Sakai

Motivation

Ray tracing is a technique to generate photo-realistic renderings based on a solid physical foundation. Because of this, it is used for various applications that rely on qualitative imagery. Furthermore, implementations are less complex than needed for rasterization when targeting for high quality. Unfortunately, ray tracing is much slower for rendering images.

Problem Statement

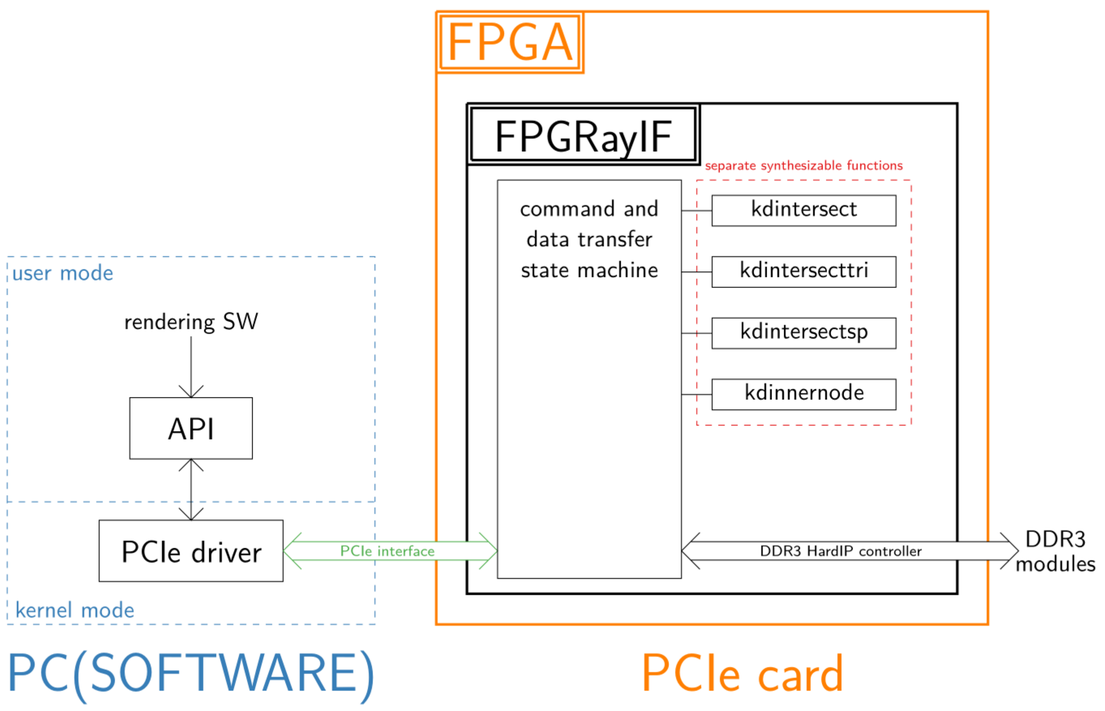
When using ray tracing algorithms to render photorealistic images, a high number of samples is needed due to the stochastic nature of those algorithms. This results a in high computational expense. Besides improvements that have been accomplished for the various algorithms, an approach for increasing the efficiency is to use hardware for accelerating existing algorithms. Previous works relied on a hardware design for the complete rendering pipeline and suffered from these problems:

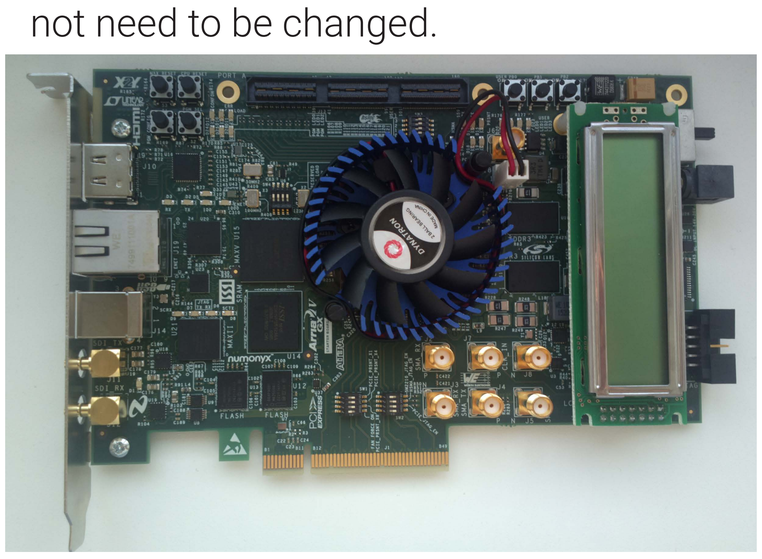
- All operations have to be provided by the design, i.e., resources of the PC are unused.
- New techniques make the hardware design obsolete.

FPGRay

Our approach provides a hardware acceleration for the k-d tree intersection algorithm through a PCIe card containing an FPGA with its own RAM to store the scene's data. The scene and rays to intersect are loaded from a PC via the PCIe interface. To access the functionality from software, a driver and C++ API for the Linux operating systems are provided. It can be used to extend the functionality of existing renderers.
FPGRay therefore combines the FPGA and the CPU for rendering, representing a hybrid solution. This solves some drawbacks of a CPU- or hardware-only renderer:

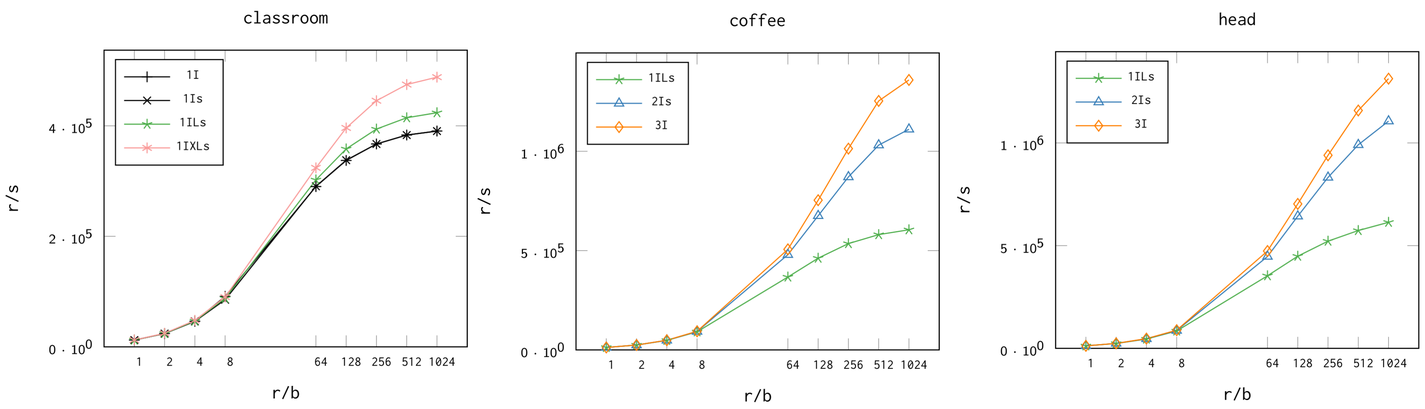
- Only functions that are frequently used and viable for hardware implementation need to be implemented.
- Seldomly used functions are processed on the CPU for higher utilization of all available resources.
- New functions can be realized as a CPU implementation, the hardware does not need to be changed.





FPGRay provides the intersection of a k-d tree through the hardware block (entity) „kdintersect". The entity supports an easy parameterization to test designs with different cache sizes, number of computational units and many more. This allows to assess the impact of the different parameters to the overall throughput and thus efficiency. Additionally, this flexible design eases the deployement on different FPGAs of different sizes for which the design can be adapted.
For verification, the renderer pbrt-v3, which provided the algorithms for kdintersect, was extended to pbrt-fpgray to use FPGRay instead of its own intersection functionality. But as hardware needs a high data rate respectively parallelization for proper utilization, a synthetic test program was written to support the intersection of multiple rays with one API call in ray batches. pbrt-fpgray can be used for generating compatible test dumps with ray batches during the rendering of a scene.

Results



- FPGRay needs a high batch size for proper utilization
- Bigger caches (see the classroom scene with three different cache sizes) increase the throughput at higher batch sizes
- Increasing the number of computational units (see the coffee and head scenes with 1, 2, or 3 traversal instances) increases the throughput too
- The overall throughput and efficiency of the currently used designs and FPGA is worse than that of CPU or GPU-only solutions
- Multiple bottlenecks and possible solutions indicate that FPGRay can surpass the poor efficiency through optimizations