FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Diplomarbeitspräsentation

GPU-based Video Processing in the Context of TV Broadcasting

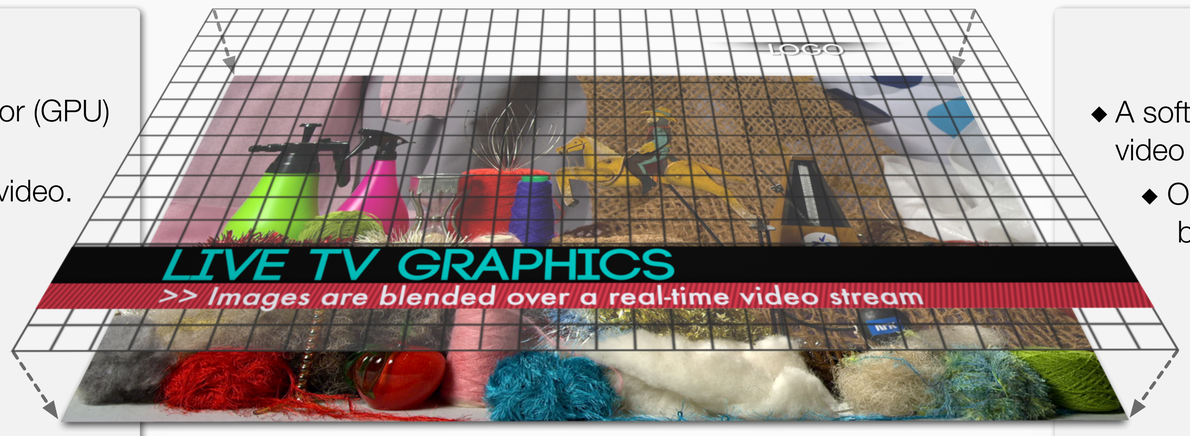
Masterstudium:
Visual Computing

Heinrich Fink

Technische Universität Wien
Institut für Computergraphik und Digitale Bildverarbeitung
Arbeitsbereich: Computergraphik
Betreuer: Assoc. Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Mitwirkung: Dipl.-Ing. Dr. Anton Fuhrmann

Motivation

- In live TV broadcasting, the graphics processor (GPU) is used to render graphics.
- A common scenario is to blend images over video.
- For rendering, video frames need to be streamed to and from video memory.
- Upcoming TV standards like UHD-1 (*4K*) result in much higher data rates of video images than previous formats.
- In order to process these data rates in real time, rendering and transfer of video need to be parallelized.



Research Questions

- Which methods can be used to parallelize the stages of an OpenGL-based video processing software?
- Which stages can be executed concurrently by hardware?
- What are the maximum data rates that can be reached and what are the limiting factors?
- How to implement GPU-based transcoding between studio-quality Y'CbCr video and linearly coded RGB?

Implementation

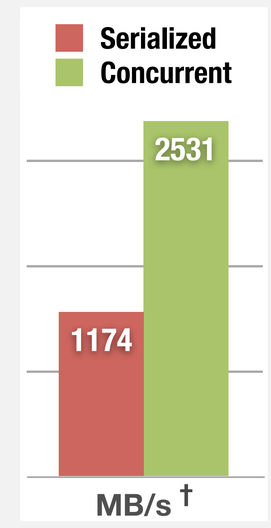
- We use the *pipeline pattern* to the build a software prototype for broadcast video processing.
- The overall algorithm is split into several thread-safe *stages.*
- We add a scheduler that can be configured to assign one or more threads to the execution of stages.
- We use OpenGL 4.x for all GPU-related tasks of the pipeline.
- We use a sophisticated communication pattern to synchronize concurrent OpenGL executions.
- We implement the Y'CbCr to RGB transcoder using different versions of GLSL and take advantage of random-writes to textures and *compute shaders*.
- We integrate a profiler into our pipeline that captures CPU-side and GPU-side execution times of stages.

Contributions

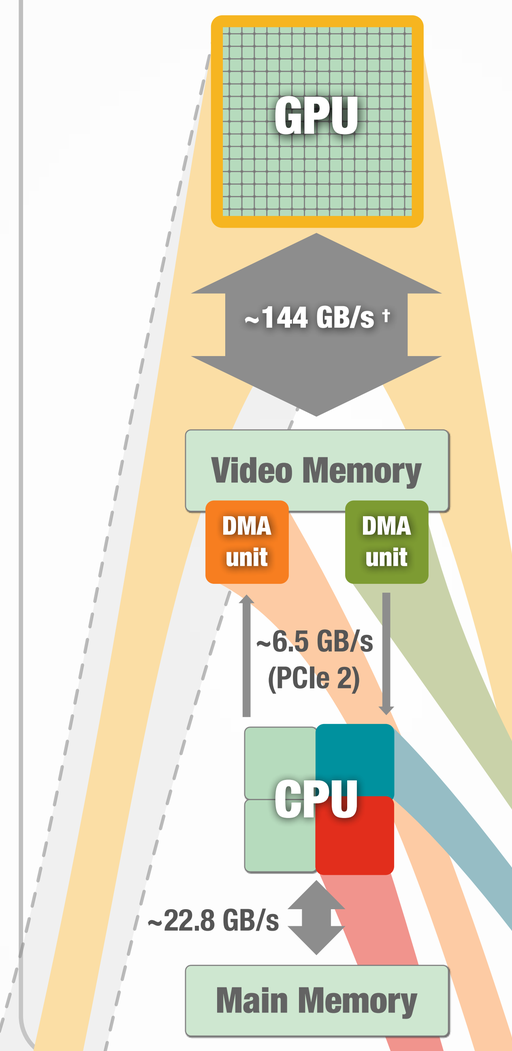
- A software model to build a highly parallelized video processing solution.
  - OpenGL-based implementation of a prototype broadcast renderer using C++11.
    - Design of a doubly-linked pipeline pattern that enables asynchronous two-way communication between stages.
    - GPU-based algorithms to transcode between 10-bit Y'CbCr and linearly coded RGB in high quality.
  - A comparison between different GLSL implementations of the transcoder module.
  - Visualization and detailed performance analysis of different configurations of our pipeline.

Results

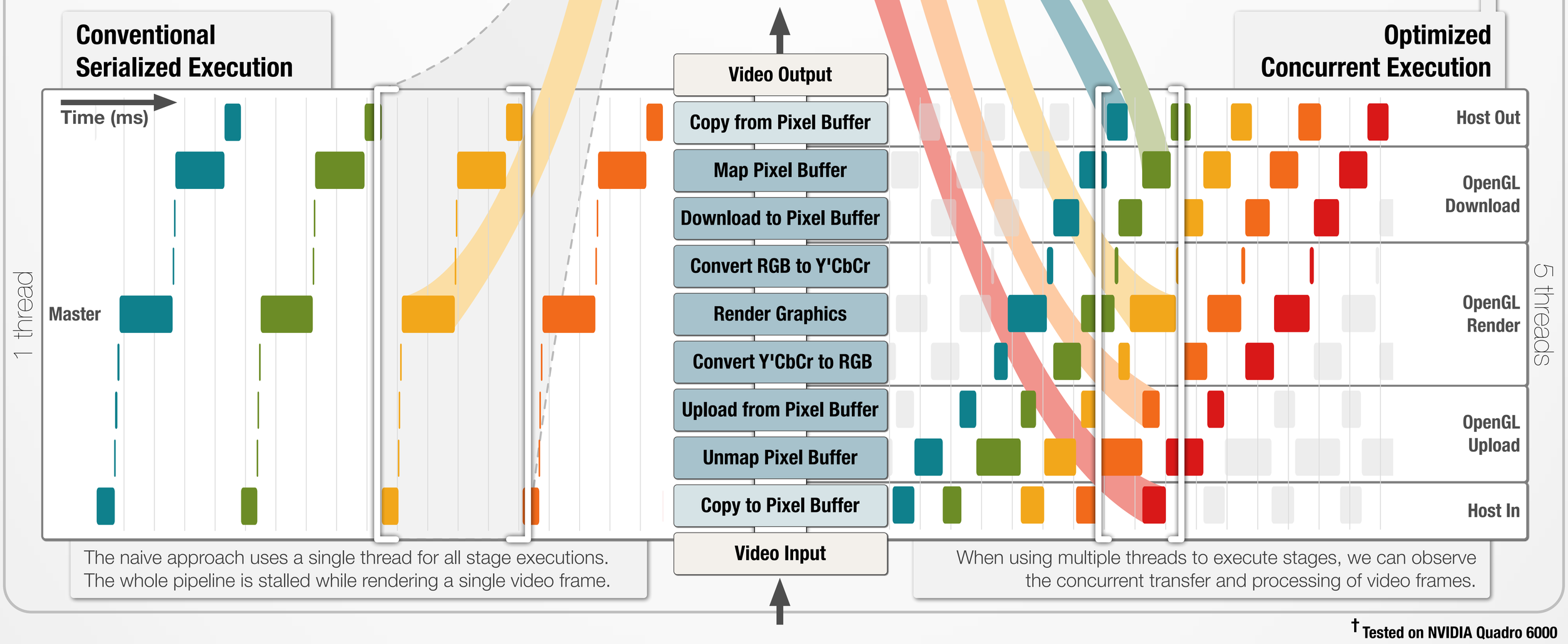
- In comparison to a conventional implementation, our pipeline enables optimizations that double the effective throughput of image data when processing high resolution video.
- This allows us to render UHD-1 in real time.
- Our profiling shows that the pipeline fully saturates the CPU memory bandwidth, i.e., we are fully taking advantage of the system's resources.
- Our GPU-based transcoder between 10-bit Y'CbCr and RGB shows better quality and is faster by an order of magnitude when compared to a state-of-the-art CPU-based conversion.
- Tests of running our prototype on different graphics devices show that each OpenGL driver implementation requires different threading strategies. The ability of our framework to configure the assignment of threads to stage executions dynamically is therefore very important.



Hardware



Software Pipeline Execution



The naive approach uses a single thread for all stage executions. The whole pipeline is stalled while rendering a single video frame.

When using multiple threads to execute stages, we can observe the concurrent transfer and processing of video frames.

† Tested on NVIDIA Quadro 6000

Kontakt: hfink@hfink.eu