VoxelCache: A Cache-Based Memory Architecture for Volume Rendering

U. Kanus, G. Wetekam, J. Hirche
WSI / GRIS
University of Tübingen

urs@gris.uni-tuebingen.de

Overview

- Motivation
- Previous work
- VoxelCache
  - Overview
  - Voxel Memory Layout
  - Block Pre-fetching
  - Cache Implementation
- Results
- Conclusion

Motivation

- High quality Volume Rendering for large datasets (>> 512^3 Voxels) still requires special purpose hardware.
- Developing ASICs is too expensive and takes too long (at least for us)
- Reconfigurable hardware devices (FPGAs) are large enough and fast enough to implement high performance graphics hardware.

Motivation

Memory is the key issue for volume rendering systems

→ need a memory system for Volume Rendering that:

- Combines regular memory access patterns and ray-casting
- No replication of volume data
- Allows for efficient caching of voxels
- Can be used with different types and arrangements of external memory
- Can be easily implemented on FPGAs

Overview

- Motivation
- Previous work
- VoxelCache
  - Overview
  - Voxel Memory Layout
  - Block Pre-fetching
  - Cache Implementation
- Results
- Conclusion

Previous Work

- VIRIM [Guenther et al. 94]:
  Eight memory modules, interleaved
- VIZARD [Knittel 97]:
  Host memory + on-board cache, lossy block compression
- VolumePro [Pfister et al. 99]:
  One memory module per pipeline, interleaved blocks (skewing), not suitable for persp. proj.
- VIZARDII [Meißner et al. 02]:
  Four memory modules + FIFOs, no on-chip cache
- Many other proposals (see paper)

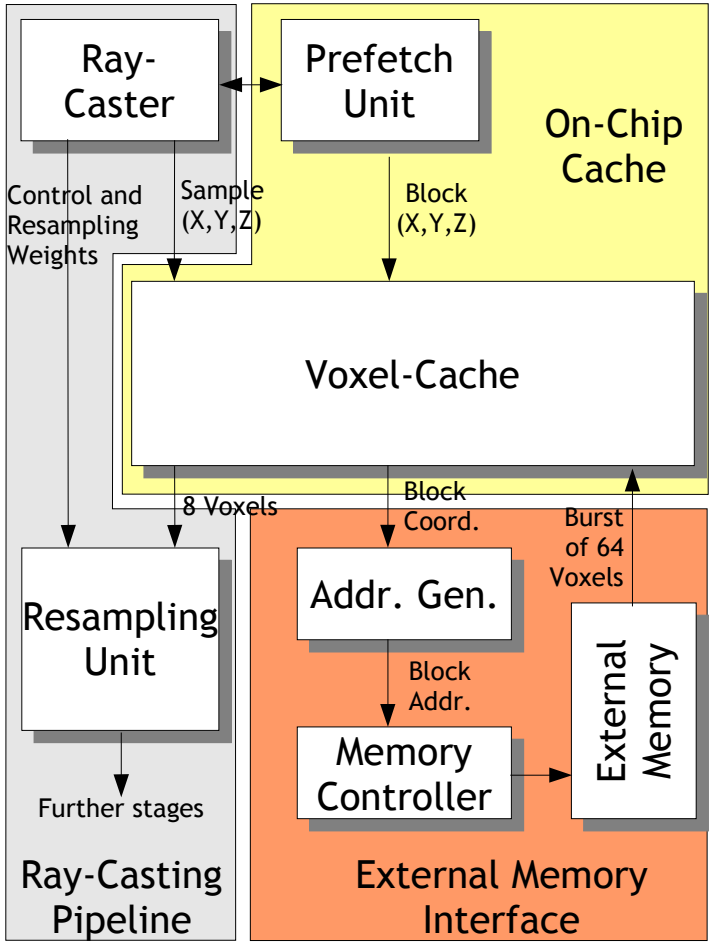
Overview

- Motivation
- Previous work
- VoxelCache
  - Overview
  - Voxel Memory Layout
  - Block Pre-fetching
  - Cache Implementation
- Results
- Conclusion

VoxelCache Overview

System split into three parts:

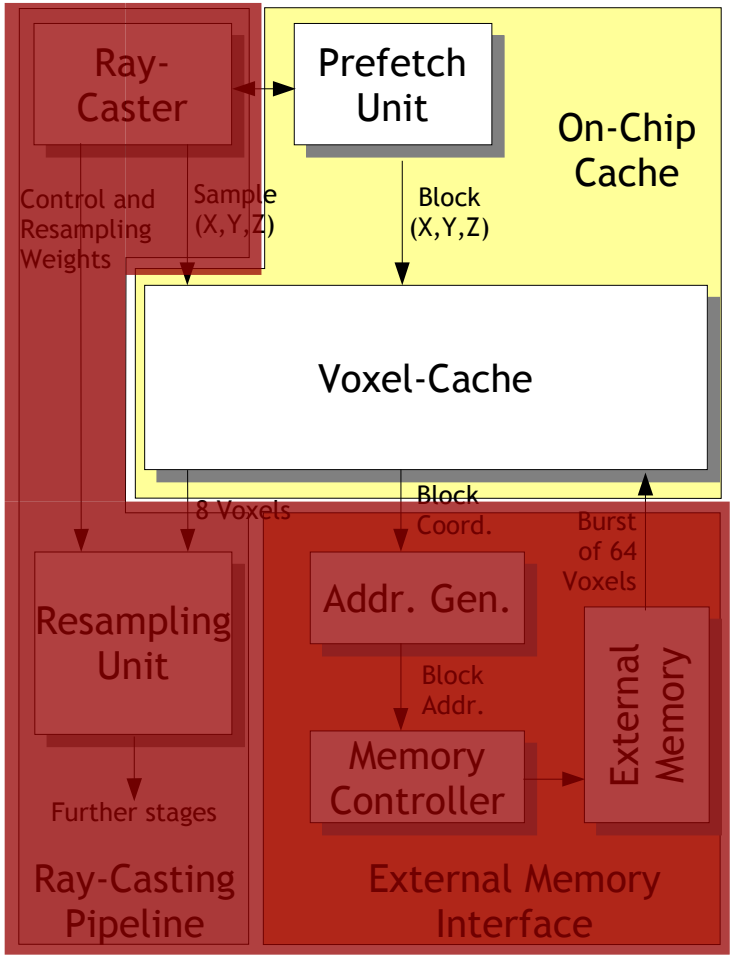
- VoxelCache on-chip Cache
- External memory interface
- Ray-casting pipeline



VoxelCache Overview

System split into three parts:

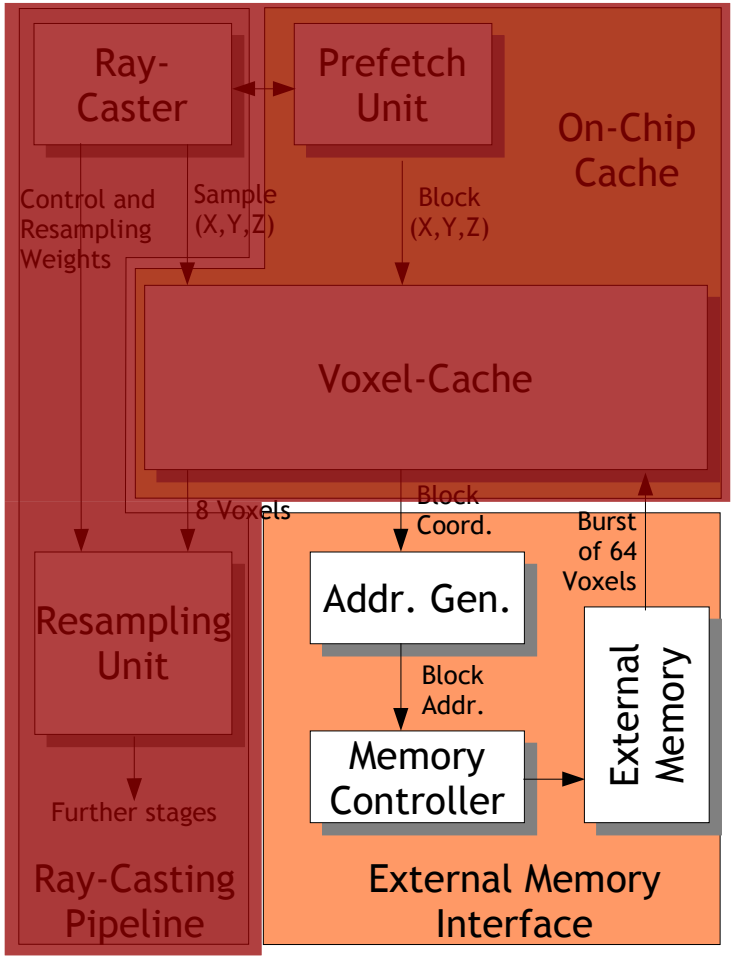
- VoxelCache on-chip Cache
  - Provides voxel neighborhood for resampling for a given sample position
  - Generic interface to external memory
  - Block prefetching to minimize cache misses
- External memory interface
- Ray-casting pipeline



VoxelCache Overview

System split into three parts:

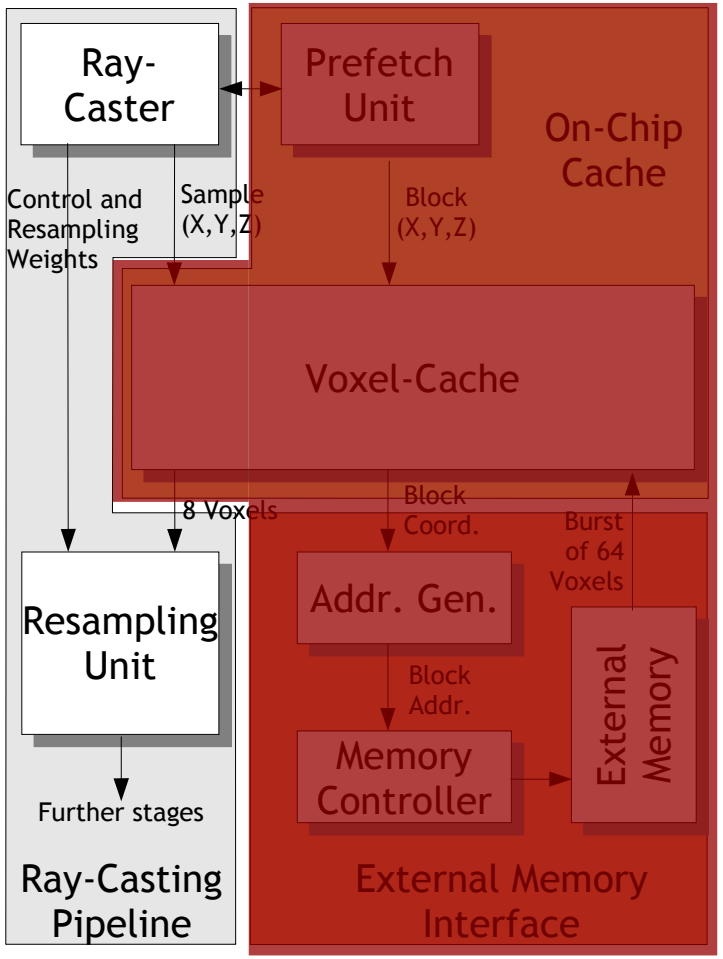
- VoxelCache on-chip Cache
- External memory interface
  - Translates block coordinates into memory addresses
  - Burst transfers of 64 voxels at a time
- Ray-casting pipeline



VoxelCache Overview

System split into three parts:

- VoxelCache on-chip Cache
- External memory interface
- Ray-casting pipeline
  - Standard ray-casting pipeline for perspective and parallel projections

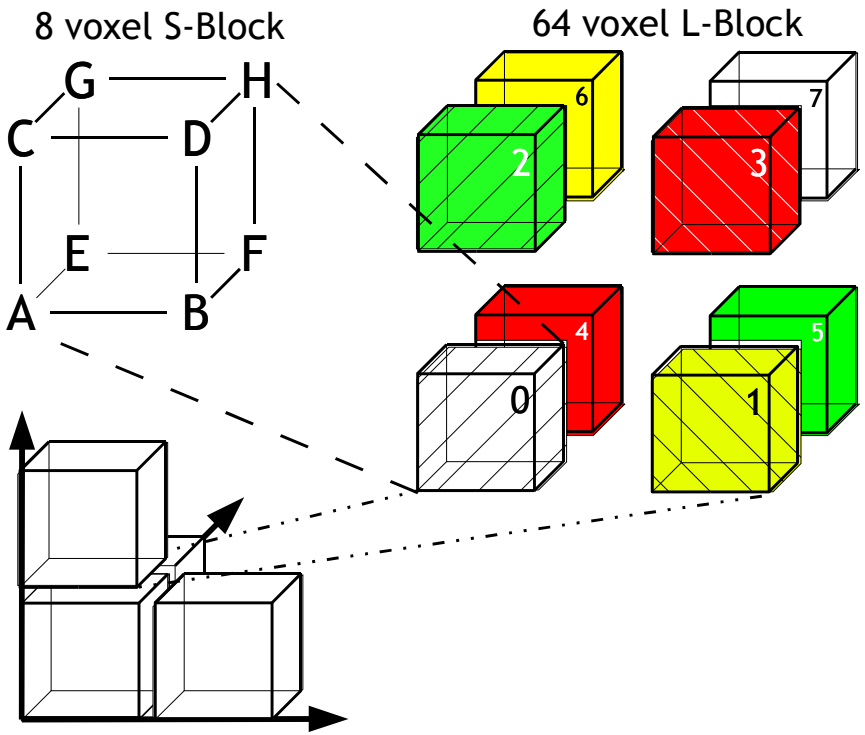


Overview

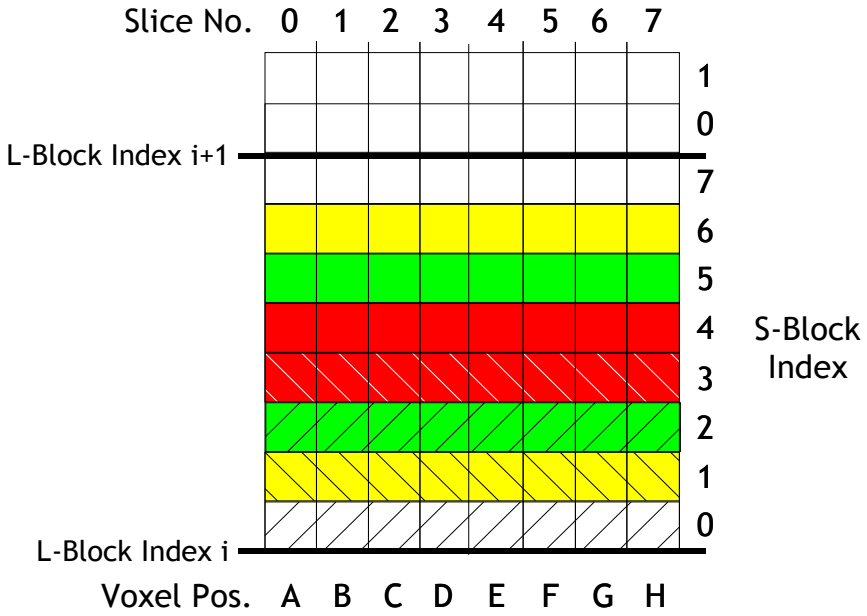
- Motivation
- Previous work
- VoxelCache
  - Overview
  - Voxel Memory Layout
  - Block Pre-fetching
  - Cache Implementation
- Results
- Conclusion

Voxel Memory Layout

Block hierarchy

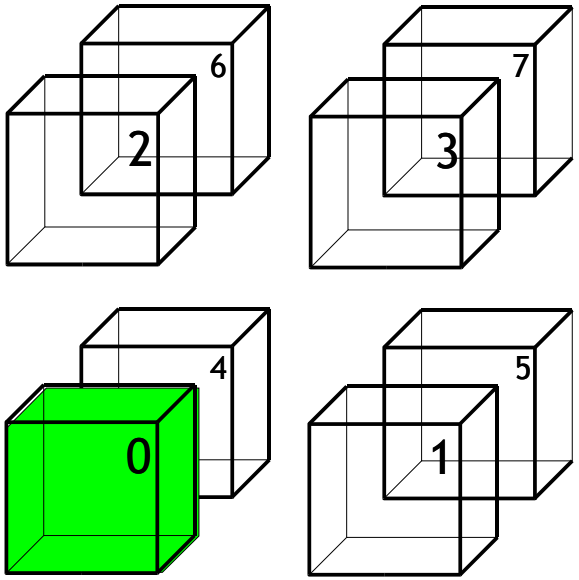


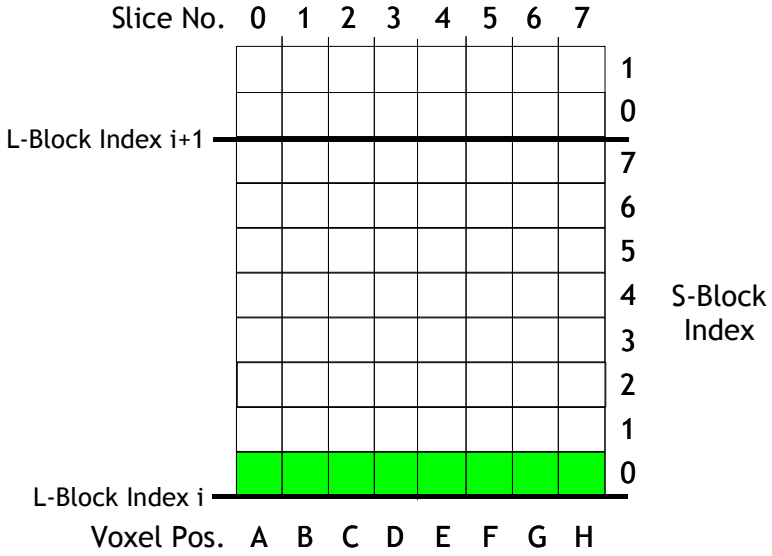
Cache memory



Addressing Voxel Neighborhoods

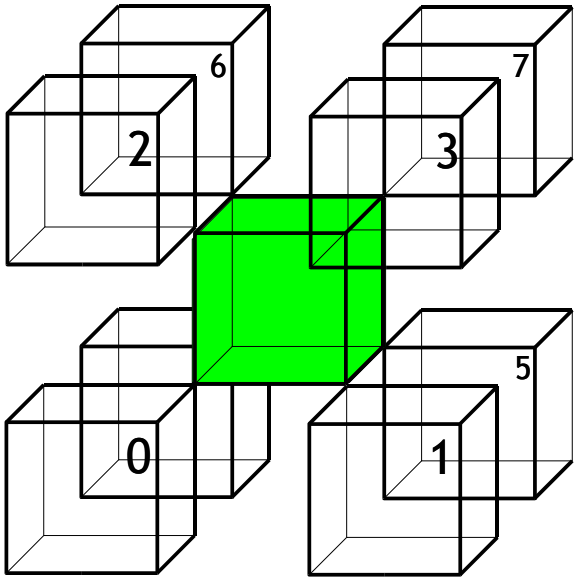
Accessing voxels from a single S-Block

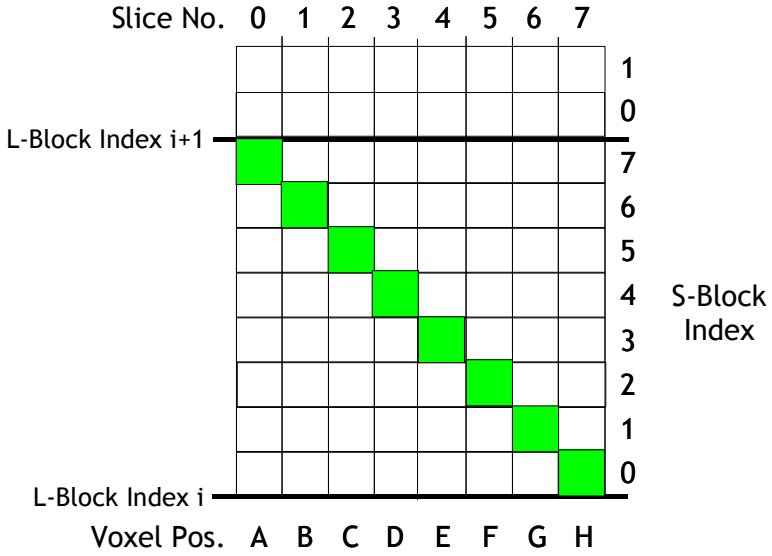




Addressing Voxel Neighborhoods

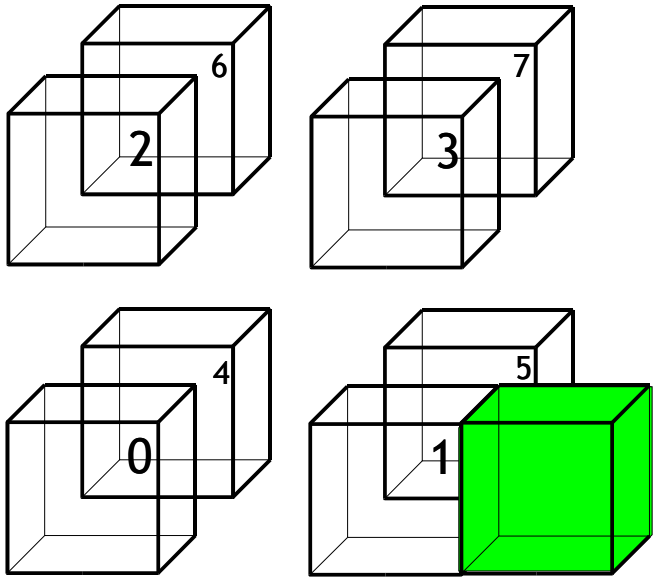
Accessing voxels from several S-Blocks

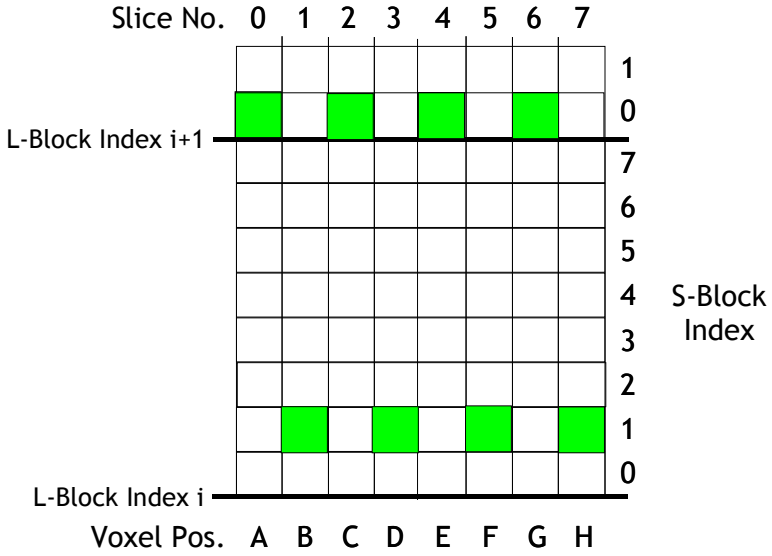




Addressing Voxel Neighborhoods

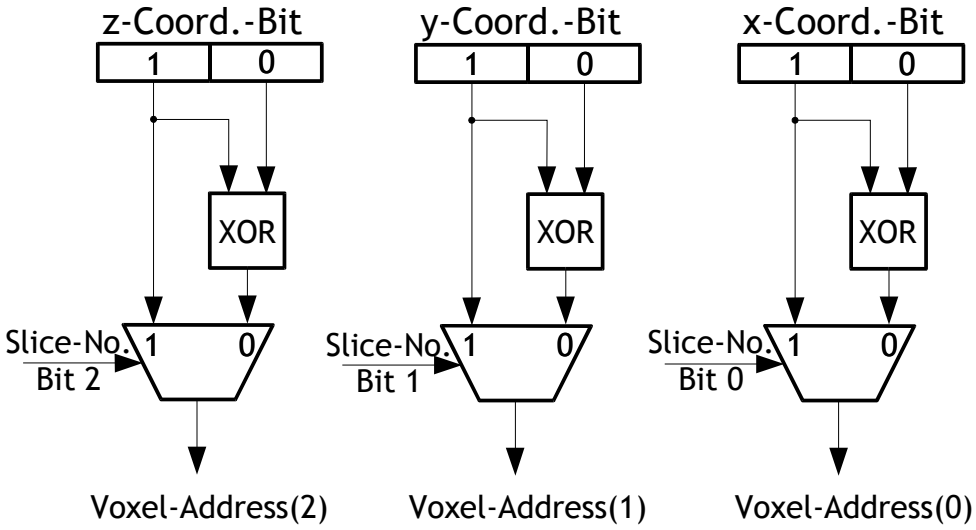
Accessing voxels from several L-Blocks





Slice Address Generator

- 3-bit voxel-address inside L-Block
- Each cache slice needs one address generator
- Simple logic (3 XOR + 3 MUX per address generator)



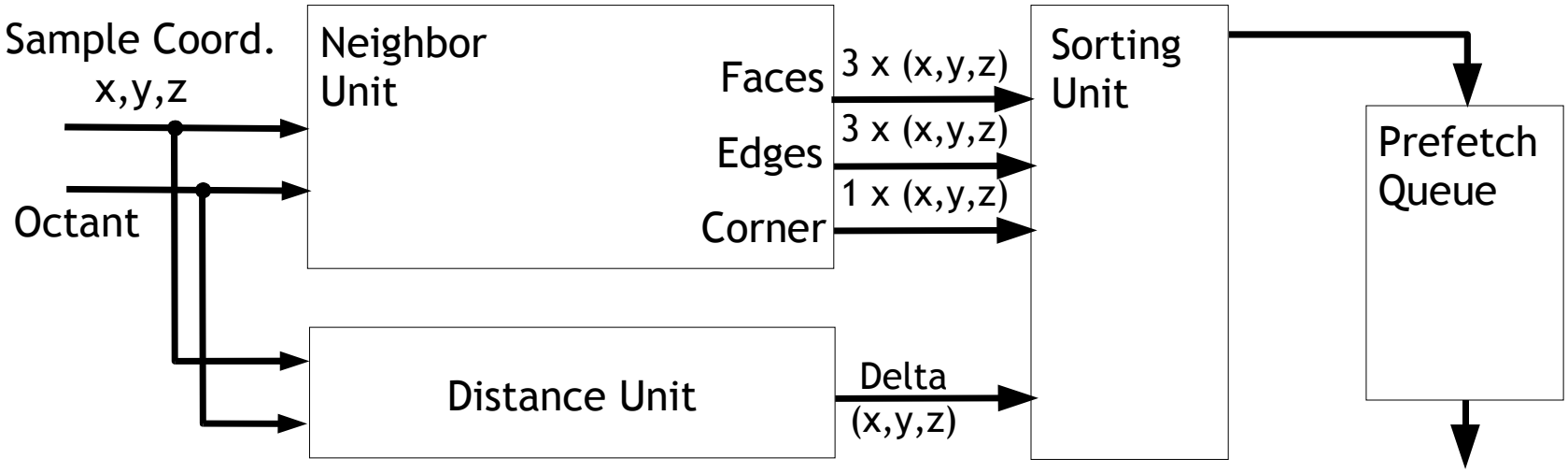
Overview

- Motivation
- Previous work
- VoxelCache
  - Overview
  - Voxel Memory Layout
  - Block Pre-fetching
  - Cache Implementation
- Results
- Conclusion

Pre-fetching Voxel Blocks

Pre-fetching of L-Blocks based on:

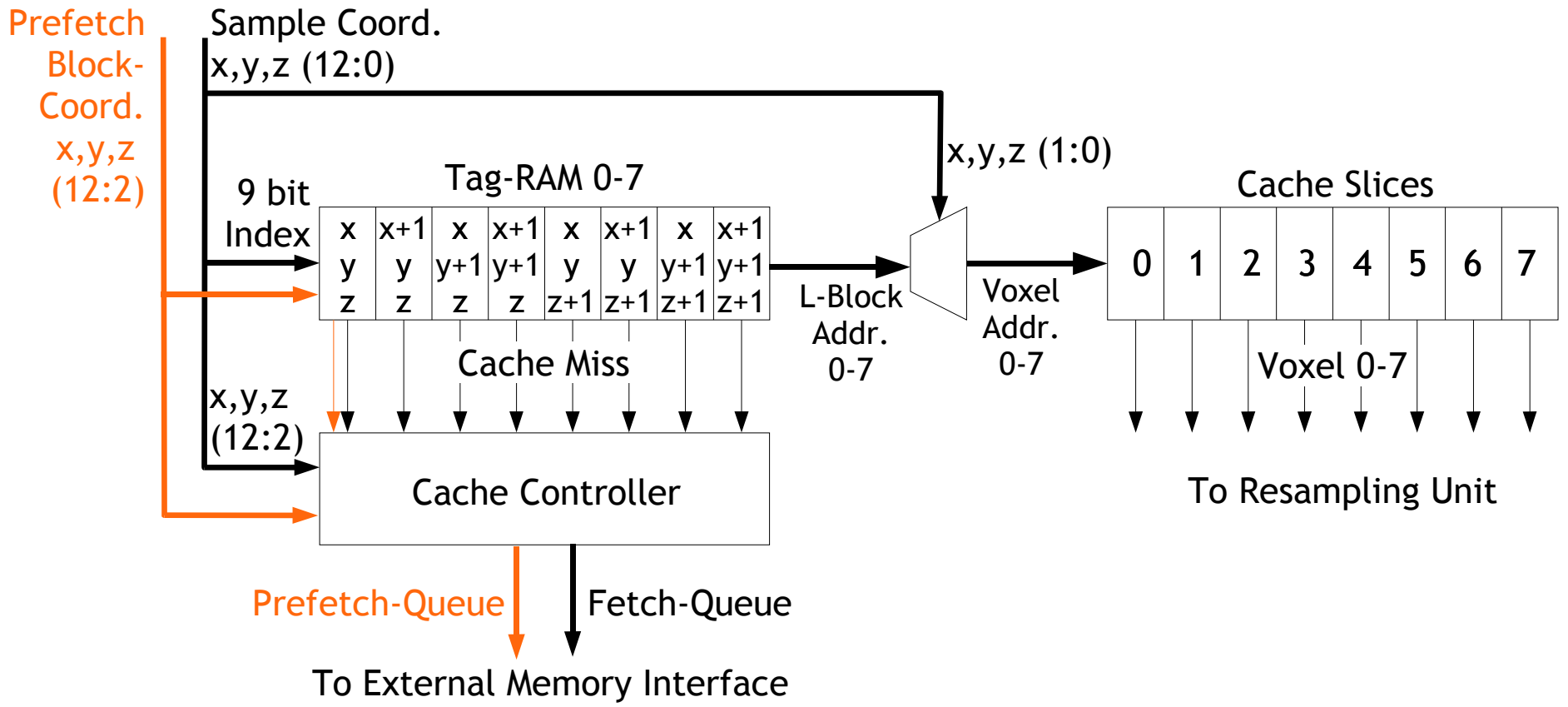
- Ray-direction (octant)
- Sample position in current L-Block



Overview

- Motivation
- Previous work
- VoxelCache
  - Overview
  - Voxel Memory Layout
  - Block Pre-fetching
  - Cache Implementation
- Results
- Conclusion

Cache Implementation



Cache Implementation

Cache characteristics:

- Tag-RAM direct mapped (9 coordinate bits)
- Arbitrary L-Block allocation in cache memory
- L-Blocks replacement in FIFO order, unless L-Block to be replaced is required for the next sampling neighborhood

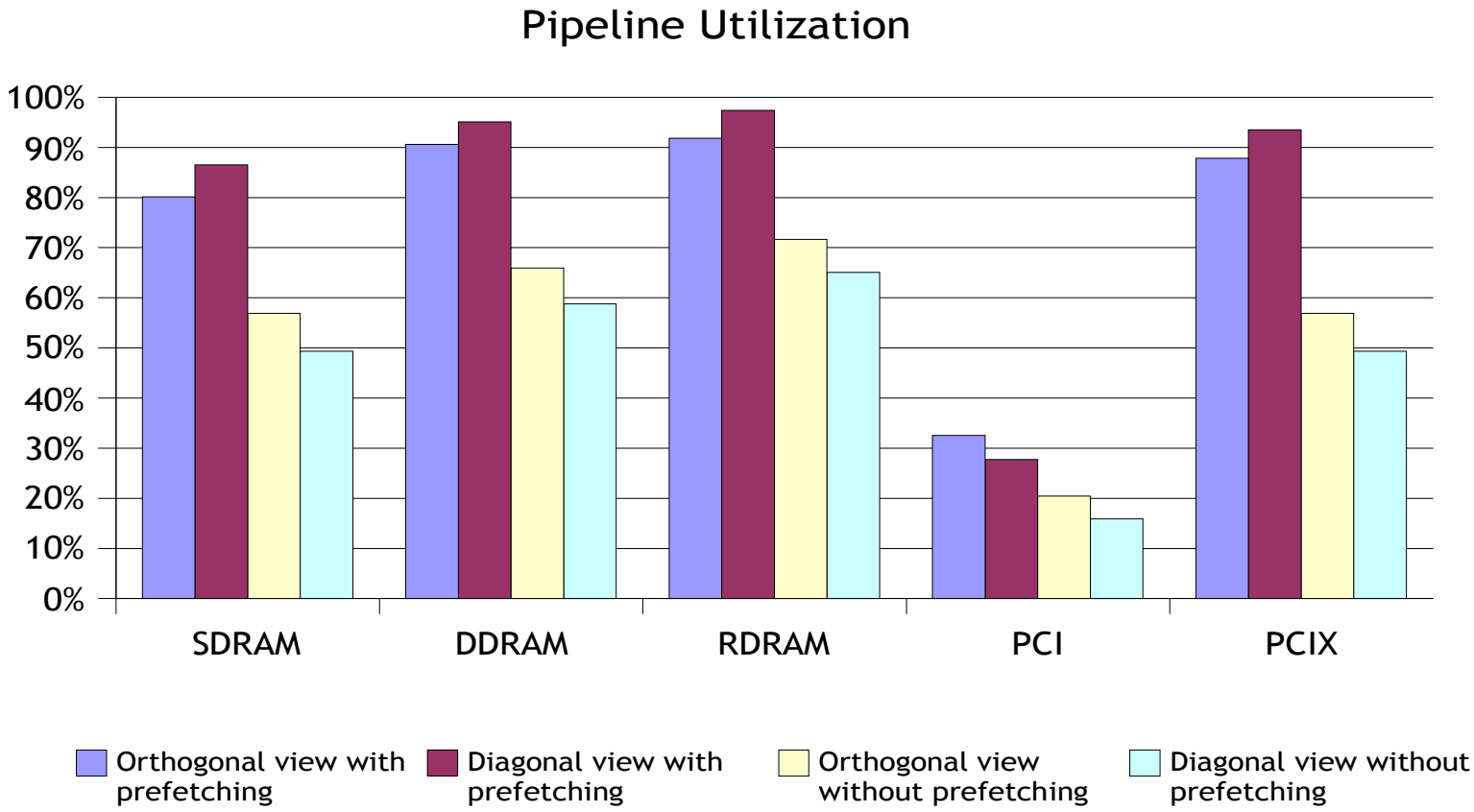
Overview

- Motivation
- Previous work
- VoxelCache
  - Overview
  - Voxel Memory Layout
  - Block Pre-fetching
  - Cache Implementation
- Results
- Conclusion

Results

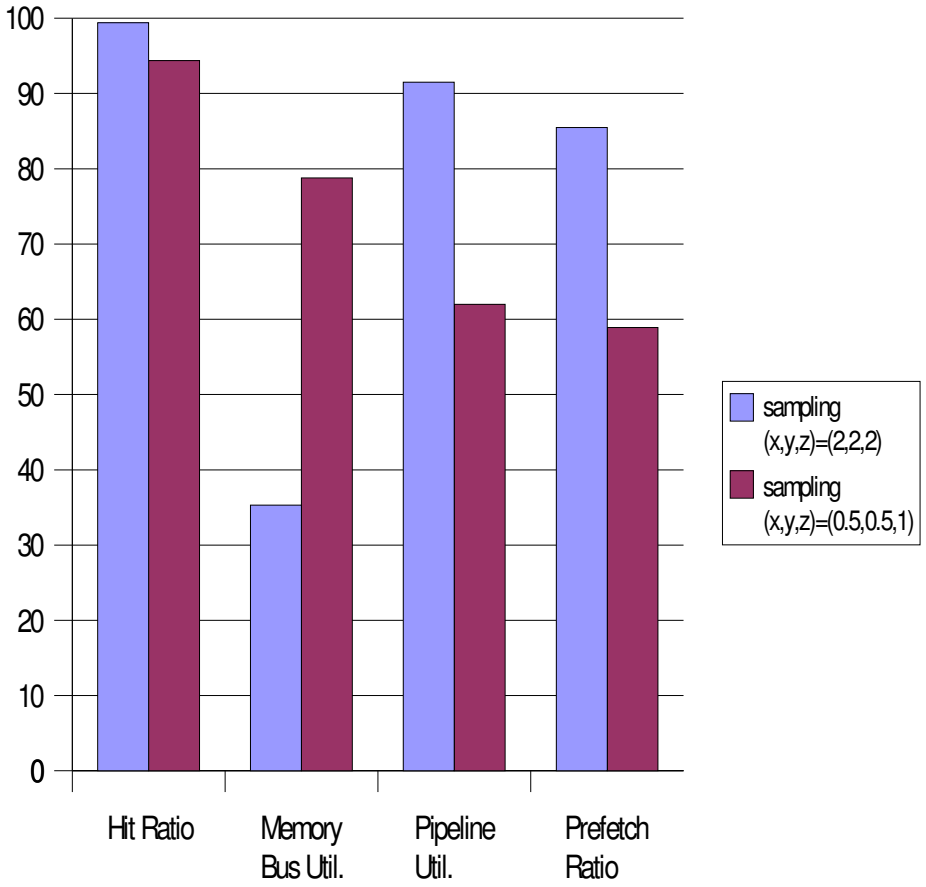
- Cycle accurate C++ simulation embedded into software ray-caster:
  - Assuming a single rendering pipeline running @ 133 Mhz
  - Different memory types defined by latency and bandwidth
  - Cache size of 128 L-Blocks
    - fits well into target FPGA architecture
  - Dataset size 136^3 voxels
    - software simulation is time-consuming
    - For size $>128^3$ an orthogonal ray doesn't fit into the cache anymore

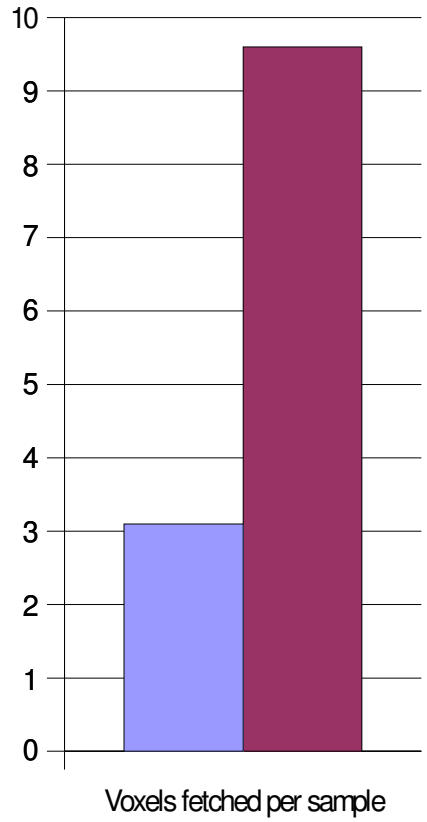
Results


Pipeline Utilization
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
SDRAM
DDRAM
RDRAM
PCI
PCIX
Orthogonal view with prefetching
Diagonal view with prefetching
Orthogonal view without prefetching
Diagonal view without prefetching

Results

Cache performance vs. Sampling (DDRAM):





Urs@ GRIS.Uni-Tuebingen.DE

Overview

- Motivation
- Previous work
- VoxelCache
  - Overview
  - Voxel Memory Layout
  - Block Pre-fetching
  - Cache Implementation
- Results
- Conclusion

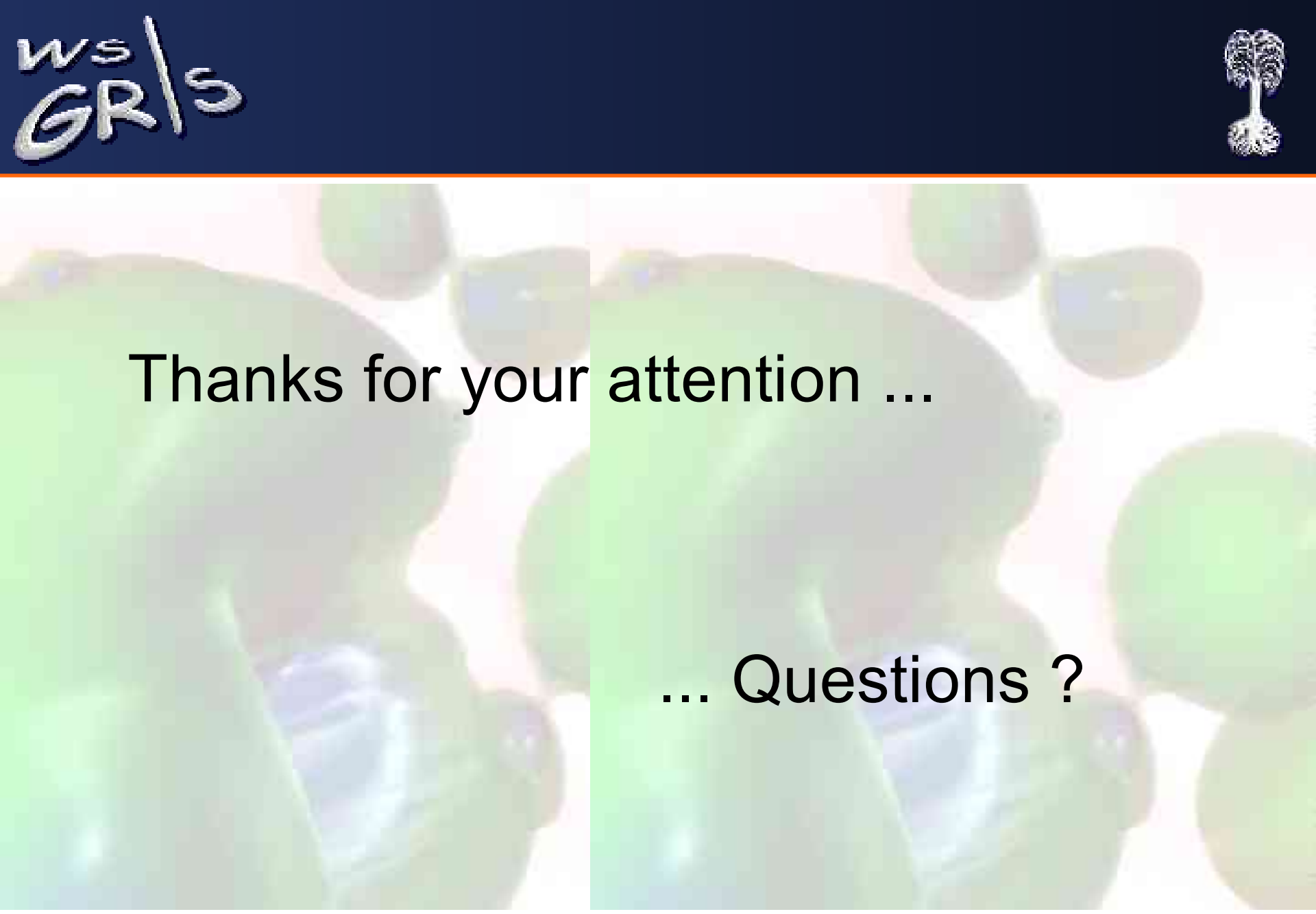
Conclusion and Outlook

- VoxelCache is an efficient memory architecture for Volume Rendering
  - Hit ratio > 98% in most cases, increases with sampling rate
  - No data replication
  - Sustained performance even for random access patterns, few pipeline idle cycles
- VoxelCache is designed for:
  - implementation on reconfigurable devices
  - easy adaption to different types of external memory

Conclusion and Outlook

Future extensions to VoxelCache:

- Integrate VoxelCache with space leaping
- Explore block compression schemes
- Efficient real-time volume data update


Thanks for your attention ...
... Questions ?