RPU: A Programmable Ray Processing Unit for Realtime Ray Tracing

Sogang University Graphics Lab.

Bong-Jun Chin

2008. 7. 22

Abstract

- This paper describes the architecture and a prototype implementation of a single chip, fully programmable Ray Processing Unit (RPU).
- 20 frames per second at 66Mhz the prototype FPGA implementation.
- SIGGRAPH 2005.



Hardware Support for Ray Tracing

- The large amount of floating point computations.
- Support for flexible control flow including recursion and branching.
- Complex memory access patterns to an often very large scene data base.

Hardware Support for Ray Tracing

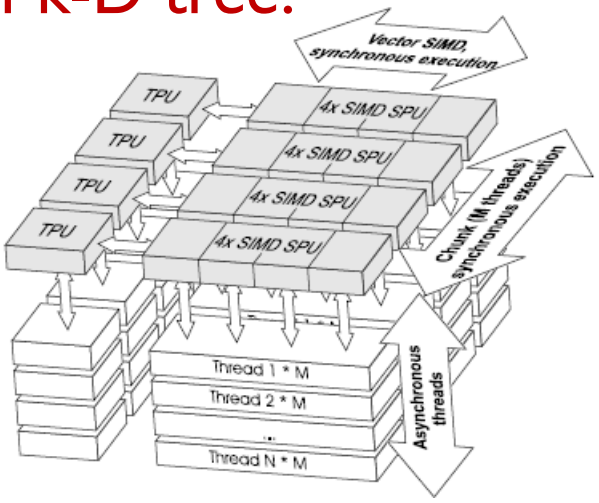
- Realtime ray tracing performace has recently been achieved even on single high-performance CPUs.
- However, higher resolutions, complex scenes and advanced rendering effects still require a cluster of CPUs for realtime performace.
- GPUs is still too limited and does not efficiently support ray tracing.
- Fully functional realtime ray tracing chip was presented by Schmittler et al. Howerever, these only support a fixed functionality and cannot be programmed.

Design Decisions

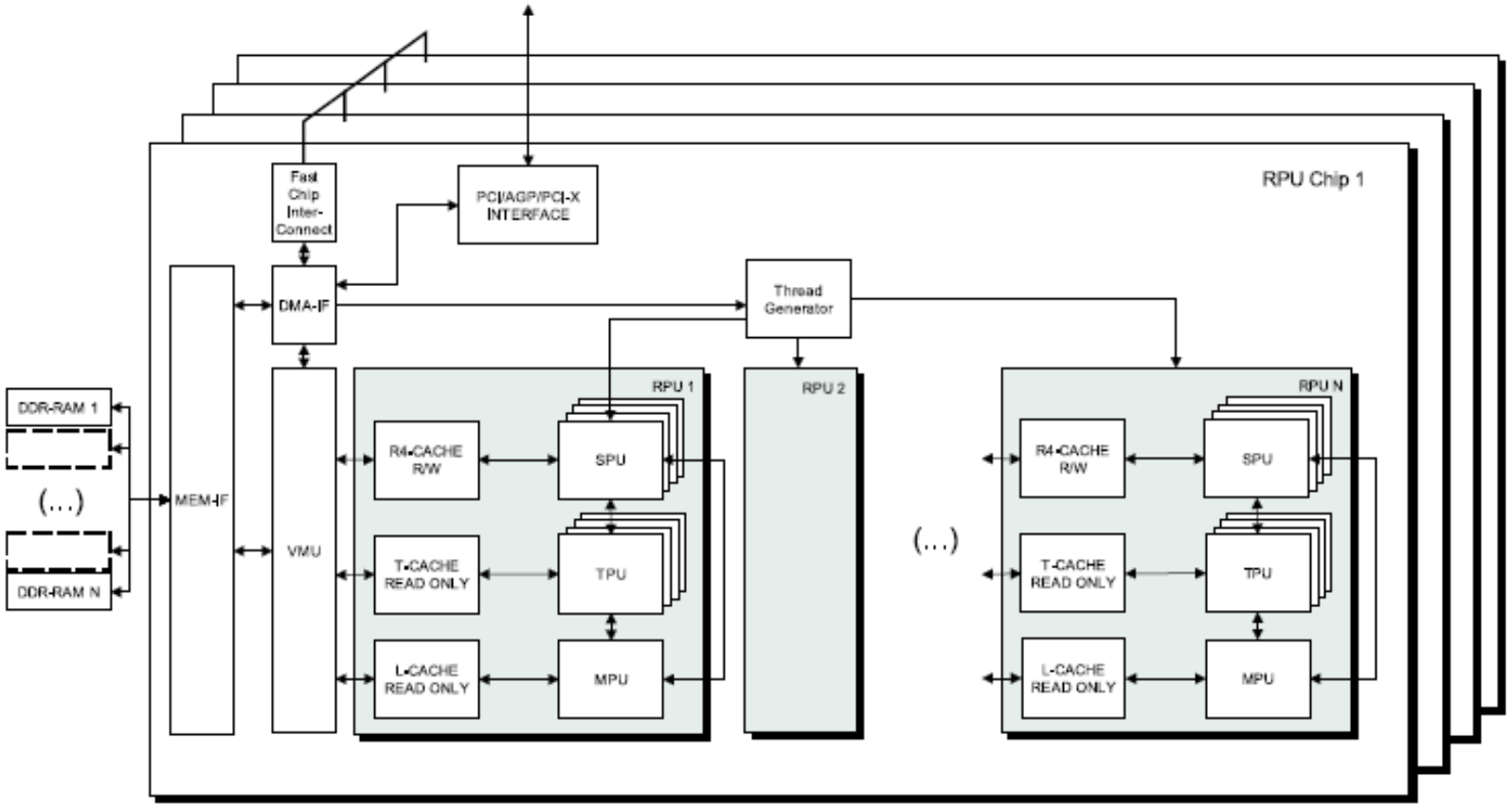
- Ray tracing is a compute intensive, recursive, and highly parallel algorithm with complex control flow requirements.
- The raw algorithm would perform a large number of mostly unstructured memory accesses, which can be greatly reduced by exploiting the coherence between rays.
- Most operations in the ray tracing algorithm are floating point vector operations, especially for shading.

Design Decisions

- Vector Operations - Shader Processing Unit (SPU)
- Threads - data parallelism in in ray tracing through a multi-threaded hardware design.
- Chunks - chunks of M threads are executed synchronously in SIMD mode in parallel by multiple SPU.
- Control Flow and Recursion – special trace instruction.
- Dedicated Traversal Units - traversal of k-D tree.



RPU Architecture



RPU Architecture

- SPU Registers – 16 register
- SPU Instruction Set – addition, multiplication, mad, …
- Pairing – vector SIMD units are utilized inefficiently for operations on scalars or too short vectors, support splitting each vector into 2/2 or 3/1 components.
- Branch Instruction
- Load Instruction
- Scheduling
- Flexible Control Flow

Programming Model

- Procedural Lighting

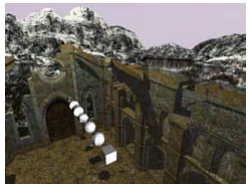
  : a global lighting shader is called that iteratively computes any incident light contribution at the point of interest.
- Procedural Geometry

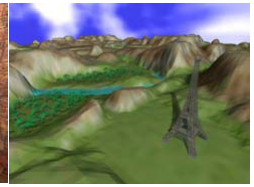
  : Geometry shaders are shaders called for any k-D tree entry encountered during traversal.
- Dynamic Scenes
- Programmable Materials

Prototype and Implementation

- Fully functional prototype of RPU architecture has been implemented using FPGA technology.
- Xilinx Virtex-II 6000-4 FPGA that is hosted on the Alpha Data ADM-XRC-II PCI-board.
- Fit a single RPU onto the FPGA chip, with four SPUs (chunk size of M = 4) and 32 concurrent hardware threads.

Results






Scene	triangles	objects	SaarCOR	RPU	OpenRT	RPU / OpenRT	RPU / SaarCOR
Scene6	806	1	44.6 fps	20.8 fps	12.9 fps	1.6	0.46
Office	34 312	1	35.9 fps	14.6 fps	10.4 fps	1.4	0.40
Quake3	39 424	1	24.6 fps	12.5 fps	11.1 fps	1.1	0.51
Quake3-p	52 790	17	19.6 fps	9.7 fps	7.9 fps	1.2	0.49
UT2003	52 479	1	18.6 fps	7.5 fps	8.0 fps	0.9	0.40
Conference	282 805	54	16.2 fps	5.5 fps	8.1 fps	0.7	0.34
Castle	20 891	8	17.5 fps	2.8 fps	9.2 fps	0.3	0.16
Terrain	10 469 866	264	11.6 fps	2.2 fps	3.5 fps	0.6	0.18
SunCOR	187 145 136	5 622	23.5 fps	4.0 fps	7.5 fps	0.5	0.17
Spheres-RT	2 spheres + 15 653	4	-	4.5 fps	-		
SPD Balls	820 spheres + 12	821	-	1.2 fps	-		

512x384 pixels, only primary ray

Conclusion and Future Work

- The RPU's programming model closely resembles that of current GPUs but extends it significantly towards general purpose computing.
- 20 frames per second at 66Mhz the prototype FPGA implementation.
- Creating spatial index structures and maintaining them across scene changes is still a challenge for highly dynamic scenes.