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 performance 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 performance.
- GPUs is still too limited and does not efficiently support ray tracing.
- Fully functional realtime ray tracing chip was presented by Schmittler et al. However, 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 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

<table>
<thead>
<tr>
<th>Scene</th>
<th>triangles</th>
<th>objects</th>
<th>SaarCOR</th>
<th>RPU</th>
<th>OpenRT</th>
<th>RPU/OpenRT</th>
<th>RPU/SaarCOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scene6</td>
<td>806</td>
<td>1</td>
<td>44.6 fps</td>
<td>20.8 fps</td>
<td>12.9 fps</td>
<td>1.6</td>
<td>0.46</td>
</tr>
<tr>
<td>Office</td>
<td>34312</td>
<td>1</td>
<td>35.9 fps</td>
<td>14.6 fps</td>
<td>10.4 fps</td>
<td>1.4</td>
<td>0.40</td>
</tr>
<tr>
<td>Quake3</td>
<td>39424</td>
<td>1</td>
<td>24.6 fps</td>
<td>12.5 fps</td>
<td>11.1 fps</td>
<td>1.1</td>
<td>0.51</td>
</tr>
<tr>
<td>Quake3-p</td>
<td>52790</td>
<td>17</td>
<td>19.6 fps</td>
<td>9.7 fps</td>
<td>7.9 fps</td>
<td>1.2</td>
<td>0.49</td>
</tr>
<tr>
<td>UT2003</td>
<td>52479</td>
<td>1</td>
<td>18.6 fps</td>
<td>7.5 fps</td>
<td>8.0 fps</td>
<td>0.9</td>
<td>0.40</td>
</tr>
<tr>
<td>Conference</td>
<td>282805</td>
<td>54</td>
<td>16.2 fps</td>
<td>5.5 fps</td>
<td>8.1 fps</td>
<td>0.7</td>
<td>0.34</td>
</tr>
<tr>
<td>Castle</td>
<td>20891</td>
<td>8</td>
<td>17.5 fps</td>
<td>2.8 fps</td>
<td>9.2 fps</td>
<td>0.3</td>
<td>0.16</td>
</tr>
<tr>
<td>Terrain</td>
<td>10469866</td>
<td>264</td>
<td>11.6 fps</td>
<td>2.2 fps</td>
<td>3.5 fps</td>
<td>0.6</td>
<td>0.18</td>
</tr>
<tr>
<td>SunCOR</td>
<td>187145136</td>
<td>5622</td>
<td>23.5 fps</td>
<td>4.0 fps</td>
<td>7.5 fps</td>
<td>0.5</td>
<td>0.17</td>
</tr>
<tr>
<td>Spheres-RT</td>
<td>2 spheres + 15653</td>
<td>4</td>
<td>-</td>
<td>4.5 fps</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPD Balls</td>
<td>820 spheres + 12</td>
<td>821</td>
<td>-</td>
<td>1.2 fps</td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.