T&I Engine: Traversal and Intersection Engine for Hardware Accelerated Ray Tracing

Jae-Ho Nah, Yonsei University
Outline

• Introduction and related work
• Overall system architecture
• Traversal with an ordered depth-first layout
• Three-phase intersection test unit
• Ray accumulation unit for latency hiding
• Simulation results and analysis
• Conclusions and future work
Outline

• Introduction and related work
• Overall system architecture
• Traversal with an ordered depth-first layout
• Three-phase intersection test unit
• Ray accumulation unit for latency hiding
• Simulation results and analysis
• Conclusions and future work
Introduction to Ray Tracing

• Rendering technique by tracing the path of light
• Generates high-quality visual effects
  • Reflection, refraction, shadows, etc.
  • Widely used for off-line rendering
• Ray tracing goes mainstream for real-time rendering [Hurley 2005; Mark 2008]
Real-time Ray Tracing

- Performance requirement [Govindaraju et al. 2008]:
  \[ 1 \text{G rays/s} \approx \frac{1024 \times 1024}{\text{resolution}} \times \frac{8}{\text{primary rays per pixel}} \times \frac{4}{\text{secondary rays per pixel}} \times \frac{25}{\text{frames per second}} \]
Related Work

- Dedicated ray tracing hardware
  - Packet-based SIMD architecture
    - SaarCOR [Schmittler et al. 2004]
    - RPU [Woop et al. 2005], D-RPU [Woop et al. 2006]
    - RTE [Davidovic et al. 2010]
  - Wide-SIMD architecture
    - StreamRay [Gribble and Ramani 2008]
  - MIMD architecture
    - TRaX [Spjut et al. 2009]
    - MIMD threaded multiprocessors [Kopta et al., 2010]
Related Work

- General purpose many-core architecture
  - Wide-SIMD
  - Larrabee [Seiler et al. 2008] → Intel MIC
  - MIMD
  - Copernicus [Govindaraju et al. 2008]
  - xPU [Mahesri et al. 2008]
  - Fermi GPU-based architecture [Aila and Karras 2010]
Related Work

• General purpose many-core architecture
  • Wide-SIMD
    • Larrabee [Seiler et al. 2008] → Intel MIC
  • MIMD
    • Copernicus [Govindaraju et al. 2008]
    • xPU [Mahesri et al. 2008]
    • Fermi GPU-based architecture [Aila and Karras 2010]

• These approaches do not yet provide sufficient performance for processing 1G rays/s for real-time distributed ray tracing
Contents

• Introduction and related work
• **Overall system architecture**
• Traversal with an ordered depth-first layout
• Three-phase intersection test unit
• Ray accumulation unit for latency hiding
• Simulation results and analysis
• Conclusions, limitations, and future work
T&I Engine

- Dedicated ray tracing hardware architecture
  - Accelerates traversal and intersection (T&I) operations

- This architecture can be integrated with existing programmable shaders
T&I Engine

- Dedicated ray tracing hardware architecture
  - Accelerates traversal and intersection (T&I) operations

- Three novel concepts
  - Traversal architecture with an ordered depth-first layout
  - Three-phase intersection architecture
  - Ray accumulation unit for latency hiding
Design Decisions

• Fixed logic design for T&I
  • High performance per area
  • Fully pipelined architecture
  • Programmable
    ray generation & shading
• Single ray tracing
  • Robust for incoherent rays
• Acceleration structure
  • $kd$-tree: best choice for single ray tracing
Overall System Architecture

- Input / output buffers
- Ray dispatcher
Overall System Architecture

- Traversal units (TRVs)
  - \(kd\)-tree traversal
Overall System Architecture

- List units (LISTs)
- Search the primitive list in a leaf node
Overall System Architecture

- The first intersection units (IST1s)
  - Ray-plane test
  - Barycentric test
Overall System Architecture

- The second intersection unit (IST2)
- Calculation of the final hit point
Overall System Architecture

- L1 and L2 caches
Outline

• Introduction and related work
• Overall system architecture
• Traversal with an ordered depth-first layout
• Three-phase intersection test unit
• Ray accumulation unit for latency hiding
• Simulation results and analysis
• Conclusions and future work
Traversal with an Ordered Depth-First Layout (ODFL)

- At each traversal step, node fetching and stack operations are required → Increase memory traffic

- We apply two methods to our architecture:
  - Ordered depth-first layout [Nah et al. 2010] (previously announced at SIGGRAPH ASIA Sketches)
  - Short-stack [Horn et al. 2007]
    - A small, $n$-entry stack to maintain the last $n$ pushes
    - Reduces the required SRAM size for stacks
Introduction to ODFL

• Goals
  • Improve the cache efficiency of depth-first layouts
  • No additional memory space

• Our approach
  • The probability of a ray intersecting with a node is proportional to its surface area [Macdonald and Booth 1990]
  • Change the arrangement criterion of child nodes: geometric position → surface area
• Child nodes are arranged by their geometric positions
  [Pharr and Humphreys 2010]
  (left node ≤ split plane ≤ right node)
Child nodes are arranged by their surface areas (SA) (left node > right node)
Tree Construction and Traversal for ODFL

• Tree construction
  • SA values are obtained by a surface area heuristic (SAH)
  • Add an 1-bit reorder flag (embedded into an 8-byte node)

• Tree Traversal
  • For front-to-back traversal, the reorder flag is referenced
Proposed Traversal Architecture

• Features
  • 1-bit NXOR operation for the ODFL
  • Supports a short-stack
  • Using pre-computed inverse direction vector

[Pharr and Humphreys 2010]
### Proposed Traversal Architecture

- **Comparisons to other architectures**

#### Table 3: Comparison to other traversal architectures. Throughput is the number of traversal steps per cycle.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FP ADD</td>
<td>4</td>
<td>4</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>FP MUL</td>
<td>0</td>
<td>4</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>FP RCP</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Stack entry</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>4</td>
</tr>
<tr>
<td>Peak throughput</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Architecture</td>
<td>SIMD</td>
<td>SIMD</td>
<td>SIMD</td>
<td>MIMD</td>
</tr>
<tr>
<td>AS</td>
<td>$k_d$-tree</td>
<td>$k_d$-tree</td>
<td>B-KD tree</td>
<td>$k_d$-tree</td>
</tr>
<tr>
<td>Special feature</td>
<td></td>
<td></td>
<td>node update</td>
<td>ODFL</td>
</tr>
</tbody>
</table>
Outline

• Introduction and related work
• Overall system architecture
• Traversal with an ordered depth-first layout
• Three-phase intersection test unit
• Ray accumulation unit for latency hiding
• Simulation results and analysis
• Conclusions and future work
Ray-Triangle Intersection Test

- Three-phase calculations
  (a) Ray-plane test

(a) Ray-plane test

Hit

Miss

\[ R(t) = O + tD \]

O : origin
D : direction

Triangle’s plane

Node’s bounding box

\[ t_{\text{min}} \]

\[ t_{\text{max}} \]

plane 1 (hit)
plane 2 (miss)
Ray-Triangle Intersection Test

- Three-phase calculations
  (b) Barycentric test

(a) Ray-plane test

Hit → Miss → Barycentric test

(b) Barycentric test

Hit → Miss

triangle 1 (hit)

triangle 2 (miss)
Ray-Triangle Intersection Test

- Three-phase calculations
  (c) Final hit point calculation

(a) Ray-plane test

(b) Barycentric test

(c) Actual hit point

- line parameter: \( t \)
- barycentric coordinate: \( u, v \)

Hit triangle
Ray-Triangle Intersection Test

- Three-phase calculations
  - (c) Final hit point calculation

  (a) Ray-plane test
  - Hit
  - Miss

  (b) Barycentric test
  - Hit
  - Miss

  (c) Actual hit point
  - Hit triangle

- Previous one-phase architectures [Schmittler et al. 2004; Woop et al. 2005; Kim et al. 2007] do not use this property

- Line parameter: $t$
- Barycentric coordinate: $u$, $v$
Three-Phase Intersection Test Unit

- IST1s perform:
  1. a ray-plane test
  2. a barycentric test

- If the ray does not pass either process, further computation and memory requests are stopped.

Case 1 (miss)
Case 2 (pass the process (a))
Hit triangle
Three-Phase Intersection Test Unit

Case 1 (miss)
Case 2 (pass the process (a))
Hit triangle

• IST1s perform
  (a) a ray-plane test
  (b) a barycentric test

• IST2 performs
  (c) final hit point calculation
Three-Phase Intersection Test Unit

 IST1s perform
 (a) a ray-plane test
 (b) a barycentric test
 IST2 performs
 (c) final hit point calculation

 Advantages
 • Reduced H/W size
 • Effective memory access
Comparison to Other Approaches

- Greatly reduced the number of arithmetic units
- High performance per area

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FP ADD</td>
<td>12</td>
<td>17</td>
<td>12</td>
<td>7</td>
<td>0.375</td>
</tr>
<tr>
<td>FP MUL</td>
<td>11</td>
<td>21</td>
<td>27</td>
<td>7</td>
<td>0.375</td>
</tr>
<tr>
<td>FP RCP</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.125</td>
</tr>
<tr>
<td>Throughput</td>
<td>0.8</td>
<td>0.5</td>
<td>1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithm</td>
<td>[Wald 2004]</td>
<td>[Möller and Trumbore 1997]</td>
<td>[Shevtsov et al. 2007]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Outline

- Introduction and related work
- Overall system architecture
- Traversal with an ordered depth-first layout
- Three-phase intersection test unit
- Ray accumulation unit for latency hiding
- Simulation results and analysis
- Conclusions and future work
Background

• Each of the T&I steps requires memory access to obtain the shape data $\rightarrow$ memory-intensive job
• The latency of off-chip memory requests can take several hundred cycles
• Need for efficient latency hiding techniques
Ray Accumulation (RA)
Unit for Latency Hiding

- Specialized hardware multi-threading for ray tracing

Ray Accumulation Unit
Input Buffer
- hit result
- cache data
- cache address

L1 CACHE

Ray Accumulation Unit
hit / miss
Buffer
- ray
- ray address
- cache data
- cache address
- cache occupation counter
- ready bit

TRV, LIST, or IST1 pipeline

hit result
cache data
cache address
L1 CACHE
ray+
shape data
Comparison to Existing H/W multi-threading

• Small working set size
  • A 32-entry RA buffer requires 4 KB of memory

• Effectively exploit temporal locality
  • The period between the ray’s shape data fetching is shorter than existing H/W multi-threading
  • This feature results from our architecture’s small working set size
Outline

• Introduction and related work
• Overall system architecture
• Traversal with an ordered depth-first layout
• Three-phase intersection test unit
• Ray accumulation unit for latency hiding
• **Simulation results and analysis**
• Conclusions and future work
S/W Setup

- Kd-tree construction:
  - SAH [Pharr and Humphreys 2010] with on-the-fly pruning [Soupikov et al. 2008]
- Sibenik, Fairy, Conference, and Hairball scenes

80 K tris  174 K tris  282 K tris  2.8 M tris
S/W Setup

• Ray type
  • primary ray, ambient occlusion (AO) ray, and diffuse ray

• 1024x768 resolution, 32 samples per ray

• This setup is the same as that in nVIDIA GPU ray tracer [Aila and Laine 2009] except for the type of acceleration structure
H/W Setup

• DRAM simulation
  • The GDDR3 memory simulator in GPGPU-Sim [Bakhoda et al. 2009]
  • 8-channel 1GHz GDDR3 memory (up to 128 GB /s)
  • First-Ready First-Come First-Serve memory access scheduling [Rixner et al. 2000]

• Cache simulation
  • L1 latency of one cycle
  • L2 latency of 20 cycles
# H/W Complexity

**Table 9:** Complexity of a T&I core measured by the number of floating-point units and the required on-chip memory.

<table>
<thead>
<tr>
<th></th>
<th>ADD</th>
<th>MUL</th>
<th>RCP</th>
<th>CMP</th>
<th>RF</th>
<th>L1 Cache</th>
<th>L2 Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 RD</td>
<td>6</td>
<td>9</td>
<td>1</td>
<td>12</td>
<td>2 KB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24 TRV</td>
<td>24</td>
<td>24</td>
<td>72</td>
<td></td>
<td>271 KB</td>
<td>192 KB</td>
<td>128 KB</td>
</tr>
<tr>
<td>6 LIST</td>
<td>56</td>
<td>56</td>
<td></td>
<td>24</td>
<td>43 KB</td>
<td>24 KB</td>
<td>32 KB</td>
</tr>
<tr>
<td>8 IST1</td>
<td>56</td>
<td>56</td>
<td></td>
<td></td>
<td>117 KB</td>
<td>128 KB</td>
<td>128 KB</td>
</tr>
<tr>
<td>1 IST2</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td>9 KB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I/O buffer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>32 KB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>86</td>
<td>92</td>
<td>2</td>
<td>108</td>
<td>476 KB</td>
<td>344 KB</td>
<td>288 KB</td>
</tr>
</tbody>
</table>
Area Estimation

- 500 MHz, 65 nm process, 200mm² die size

![Table 10: Area estimates of a T&I core.](image)
Cycle-Accurate Simulation Results

- Assumption: four T&I cores (48.50 mm²)
Comparison to Other Approaches

- Diffuse path tracing
- Conference scene
Outline

- Introduction and related work
- Overall system architecture
- Traversal with an ordered depth-first layout
- Three-phase intersection test unit
- Ray accumulation unit for latency hiding
- Simulation results and analysis
- Conclusions and future work
Conclusions and Future Work

• A novel hardware architecture for tree traversal and intersection tests

• Future work
  • Support various primitive types
  • Support dynamic scenes
  • Combine our architecture with shading filter stacks [Gribble and Ramani 2008] for complex shading
  • ASIC verification
Acknowledgement & Q&A

• This work was supported by Samsung Electronics Co., Ltd.

• Any questions?