RayChip®: Real-time Ray-tracing Chip for Embedded Applications

Woo-Chan Park¹,², Hee-Jin Shin¹, Byoungok Lee¹, Hyungmin Yoon¹, and Tack-Don Han¹,³

¹Siliconarts, Inc., ²Sejong University, ³Yonsei University

heejinshin@siliconarts.com

Copyright 2014 Siliconarts, Inc. All rights reserved
Table of Contents

- Introduction
- Ray-tracing Algorithm
- RayChip® Series 1000
- RayCore® and RayTree®
- Performance
- Summary
- Future Plan
- Q&A
**Introduction**

- Ray-tracing is a classic global illumination algorithm for photo-realistic rendering, however, it requires tremendous computing power to create high-quality images.
- RayChip® is the world’s first commercialized chip targeted to realize real-time ray tracing for embedded applications.
- This chip provides sufficient performance for real-time ray tracing, a diverse set of graphics functionalities, and easy-to-use RayCore® API.

**Local Illumination**
- OpenGL ES 1.0/1.1
- OpenGL ES 2.0/3.0

**Global Illumination**
- Basic Ray-tracing
- Advanced Ray-tracing
- Distribution Ray-tracing & Indirect Illumination*

- Color impression on objects
- Modifiable object color, modifiable imagery pixel color
- Major effect of light – reflection, refraction, transmission, and shadow
- Realistic shadow and curvature by calculating volume of light
- Diffused and irregular reflection, distance and camera effect

* Movie-quality graphics
Ray-tracing Algorithm – Fundamental

- Ray-tracing generates an image by tracing the path of light through pixels in an image plane and simulating the effects of its encounters with virtual object.

- **Advantages of Ray-tracing algorithm [1]**
  - Supporting global illumination effects such as reflection, refraction, shadow, transmission
  - Less computational complexity in object numbers (e.g., O (log N))
  - On-demand computation
  - Declarative scene description
  - Parallel (as nature)

- **Disadvantages of Ray-tracing algorithm [1]**
  - Tremendous computation (e.g., traversal and intersection process)
  - Problem in supporting fully dynamic scene (e.g., O(N log N))
  - Difficult to map ray-tracing algorithm in streaming framework
  - High memory bandwidth

Ray-tracing Algorithm – Ray-tracing VS. Rasterization

Ray-tracing

- Ray Generation
- Traversal
- Intersection Test
- Shading
- Texture Mapping

Rasterization

- Geometry Processing
- Scan Conversion
- Texture Mapping
- Depth Test
- Color Blending

S: shadow
T: transparency
R1: refraction
R2: reflection
Ray-tracing Algorithm – Advantages of Ray-tracing

- **Easy to create**
  - Simulates effects of light automatically: natural shadow, reflection, refraction, and transmission of lights (Reduced workload to develop light-related artifacts)
  - Even novice designer is able to develop 3D graphic contents without much difficulty

- **Cost-effective**
  - Tremendously reduced cost to develop 3D graphic contents
  - Spread out of reality-like 3D graphics UI and applications

3D graphics model data is developed by a graphic designer using SW authoring tools such as 3ds Max or Maya.
RayChip® Series 1000
RayChip® Series 1000 – Contribution

- The world’s first real-time ray-tracing chip for embedded applications
  - Full hard-wired logic to achieve real-time performance for ray-tracing rendering
  - Processes multiple ray bounces recursively to create realistic images
  - Maintains high throughput pipeline by adopting MIMD parallel architecture to trace individual rays
  - Scalable architecture based on tile scheduling

- High performance acceleration structure (AS)-building HW of dynamic scenes
  - Real-time ray tracing requires per frame AS building in dynamic scenes
  - RayTree®, an AS-building HW satisfies the following challenging goals:
    - Fast $kd$-tree build while maintaining high tree quality
    - Minimized memory access
    - Exploitation of burst memory access

- Easy-to-use OpenGL ES-familiar API support
  - Provides a diverse set of graphics functionalities and an OpenGL ES 1.1-familiar API
  - Allows developers to create high-quality 3D graphics applications at lower cost
RayChip® includes 6 cores of RayCore® IP and 1 core of RayTree® IP to provide high performance ray-tracing rendering up to HD resolution, 60fps

- ARM11 CPU, HDMI1.3x, USB2.0, and other peripherals are added
## RayChip® Series 1000 – Specifications

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part No.</td>
<td>SG141F</td>
</tr>
<tr>
<td>Technology</td>
<td>Fujitsu 55nm low-power technology</td>
</tr>
<tr>
<td>Die area</td>
<td>9.6 × 9.4 mm²</td>
</tr>
<tr>
<td>Package</td>
<td>17 × 17 mm, 400 FBGA</td>
</tr>
<tr>
<td>Voltage</td>
<td>Core 1.2V, I/O 1.8V, 3.3V</td>
</tr>
</tbody>
</table>

### Key Components
- RayCore®
- RayTree®
- ARM1176JZF-S
- DDR-3, USB 2.0, HDMI 1.3, SDIO, 2D Engine, System Bus (AXI), Peripherals

### RayCore®
- Six-core real-time ray-tracing GPU
- 30M gate counts
- 0.85 W/core, Max. clock 266 MHz
- Performance: 100M rays/s (MRPS), 60FPS

### RayTree®
- One scan-tree unit / two \(kd\)-tree units
- 3.5M gate counts, Max. clock 266MHz
- Performance: 1M triangles/s
**RayChip® Series 1000 – Process Flow**

<table>
<thead>
<tr>
<th>Pre processing</th>
<th>GPU Rendering (Ray-tracing)</th>
<th>Post processing</th>
<th>Display processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D Data Loading</td>
<td>Scene Management</td>
<td>Acceleration Structure Building</td>
<td>RayChips® Process Flow Chart</td>
</tr>
<tr>
<td>Scene Management</td>
<td>Ray Generation</td>
<td>Traversal &amp; Intersection Test</td>
<td>AXI ARM11 CPU</td>
</tr>
<tr>
<td>Acceleration Structure Building</td>
<td>Ray Generation</td>
<td>Traversal &amp; Intersection Test</td>
<td>AXI RayTree®</td>
</tr>
<tr>
<td>Ray Generation</td>
<td>Ray Generation</td>
<td>Traversal &amp; Intersection Test</td>
<td>AXI RayCore® 6x</td>
</tr>
<tr>
<td>Traversal &amp; Intersection Test</td>
<td>Shading</td>
<td>Shading</td>
<td>AXI 2D Engine</td>
</tr>
<tr>
<td>Shading</td>
<td>Shading</td>
<td>Texture Mapping</td>
<td>HDMI</td>
</tr>
<tr>
<td>Texture Mapping</td>
<td>Texture Mapping</td>
<td>Texture Mapping</td>
<td>Display Processor</td>
</tr>
<tr>
<td>Texture Mapping</td>
<td>2D Operation (BitBLT, Scaling, PIP)</td>
<td>Ray-traced images</td>
<td>Display Processing</td>
</tr>
<tr>
<td>2D Operation (BitBLT, Scaling, PIP)</td>
<td>Ray-traced images</td>
<td>Final images</td>
<td>Final images</td>
</tr>
<tr>
<td>Ray-traced images</td>
<td>Final images</td>
<td>Final images</td>
<td>Final images</td>
</tr>
<tr>
<td>Final images</td>
<td>Final images</td>
<td>Final images</td>
<td>Final images</td>
</tr>
<tr>
<td>Final images</td>
<td>Final images</td>
<td>Final images</td>
<td>Final images</td>
</tr>
</tbody>
</table>

Siliconarts RayChips® Presentation, Aug. 12, 2014 @ Hot Chips 2014
RayChip® Series 1000 – Process Flow

- Tile scheduling: each RayCore® renders a tiled image one at a time
  - RayCore#1 renders tiled image#1, RayCore#2 renders tiled image#2, and so on

- Process order may change due to delay in certain RayCore® rendering
  - RayCore#2 takes over RayCore#1’s task (e.g., tiled image#7) to efficiently process ray-tracing

![RayChip® Diagram with RayCore#1 to #6 and Data Memory]
RayChip® Series 1000 – RayCore® API

- Easy-to-use API for ray-tracing content development
  - Supports interface to develop ray-tracing 3D contents
  - Consists of API libraries similar to OpenGL ES 1.1 with ray-tracing specific functions

- Complete API specifications and ray-tracing programming guide are available on Siliconarts’ website (www.siliconarts.com)

```c
// set a box
rcStaticSceneBegin();
...
rcVertexPointer(3, RC_FLOAT, 0, box);
rcGenMaterials(1, &material_box);
rcBindMaterial(material_box);
rcMaterialf (RC_FRONT_AND_BACK, RC_REFLECTION, 0.0f);
rcMaterialfv(RC_FRONT_AND_BACK, RC_DIFFUSE, &cyan.r);
rcDrawArrays(RC_TRIANGLES, 0, 30);
rcDisableClientState(RC_VERTEX_ARRAY);
rcStaticSceneEnd();
```

<RayCore® API Library>  <RayCore® API Flow Chart>
RayChip® Series 1000 – Target Application

- Content-rich applications in TV/STB, digital signage and dongle mini PC
  - Stand-alone ray-tracing enabling device

End Product | RayChip® Value Proposition | Application Example
---|---|---
Consumer TV | Low power, dynamic 3D UI | “Edutainment TV Dongle”
| Intuitive, easy-to-use UX | Natural expression of high-quality 3D graphic content
Commercial TV / Digital Signage | Virtual advertisement | Maximizes user concentration
| Augmented reality (AR) | Improves learning ability
Media Box / Dongle Mini PC | Edutainment content with intuitive 3D UI and game | Interactive and easy-to-control UX
| | | Game Console / Arcade Machine
Game Console / Arcade Machine | Immersive 3D game |
RayCore® and RayTree®
RayCore® – Design Decisions

- **Fixed-pipeline architecture**
  - Fully-hardwired pipeline approach for high area and power efficiency
  - GPU in modern mobile and embedded AP can be combined for shader programming

- **MIMD vs. SIMD**
  - MIMD architecture allows higher HW utilization regardless of ray coherence

- **Unified Traversal & Intersection (‘T&I’) units vs. separate T&I units**
  - Unified T&I units perform T&I operations in a single pipeline
  - Load imbalance problem eliminated in prior separate T&I units

- “Looping for the next chance”: Efficient memory latency hide technique
  - Effectively provides HW multi-threading to hide memory latency due to cache misses

- **Acceleration structure (‘AS’) build unit**
  - *kd*-tree AS produces faster traversal and better cache efficiency
  - Dedicated *kd*-tree build HW makes it possible to meet long tree-build time
RayCore® – RayCore® Architecture

- **Setup-processing unit**
  - Passes ray information to ray-generation

- **Ray-generation unit**
  - Primary/secondary ray generation

- **T&I units**
  - Node traversals
  - Ray-triangle intersection test

- **Hit-point calculation unit**
  - Calculate the position (x,y,z) of the hit point

- **Shading unit**
  - Phong illumination
  - Texture mapping
  - Inverse displacement mapping

<RayCore® Architecture>
RayCore® – MIMD T&I Unit

- MIMD architecture is more efficient in implementing real-time ray tracing
  - MIMD has six to ten times higher performance than SIMD in a similar die area [2]

### <Unified T&I Pipeline>

<table>
<thead>
<tr>
<th>Category</th>
<th>SIMD</th>
<th>MIMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pros</td>
<td>Memory bandwidth reduction</td>
<td>Ideally perfect utilization</td>
</tr>
<tr>
<td>Cons</td>
<td>Low utilization rate in case of incoherent rays</td>
<td>Expensive HW cost</td>
</tr>
<tr>
<td>Performance</td>
<td>Good only in the case of coherent rays</td>
<td>Best (with L2 cache)</td>
</tr>
</tbody>
</table>

### <MIMD vs. SIMD Pros and Cons>

<table>
<thead>
<tr>
<th>MIMD</th>
<th>Ray Type</th>
<th>MIMD Issue Rate</th>
<th>MIMD MRPS</th>
<th>SIMD</th>
<th>Ray Type</th>
<th>GTX SRM eff.</th>
<th>GTX MRPS</th>
<th>GTX</th>
<th>Ray Type</th>
<th>GTX SRM eff.</th>
<th>GTX MRPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>147mm²</td>
<td>Primary</td>
<td>74%</td>
<td>376</td>
<td>70%</td>
<td>369</td>
<td>76%</td>
<td>74%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Diffuse</td>
<td>53%</td>
<td>286</td>
<td>57%</td>
<td>330</td>
<td>37%</td>
<td>53%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>175mm²</td>
<td>Primary</td>
<td>77%</td>
<td>387</td>
<td>73%</td>
<td>421</td>
<td>79%</td>
<td>77%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Diffuse</td>
<td>67%</td>
<td>355</td>
<td>70%</td>
<td>402</td>
<td>46%</td>
<td>67%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIMD</td>
<td>Ray Type</td>
<td>GTX SRM eff.</td>
<td>GTX MRPS</td>
<td>GTX SRM eff.</td>
<td>GTX MRPS</td>
<td>GTX SRM eff.</td>
<td>GTX MRPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GTX285</td>
<td>Primary</td>
<td>74%</td>
<td>142</td>
<td>76%</td>
<td>75</td>
<td>77%</td>
<td>142</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Diffuse</td>
<td>46%</td>
<td>41</td>
<td>46%</td>
<td>41</td>
<td>49%</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MIMD MRPS/mm² ranges from 2.56 (Conference, primary rays) to 0.73 (Sibenik, diffuse rays) for both configs
SIMD MRPS/mm² ranges from 0.25 (Conference, primary rays) to 0.07 (Fairy, diffuse rays)
SIMD (no texture area) MRPS/mm² ranges from 0.47 (Conference, primary) to 0.14 (Fairy, diffuse)

“Looping for the next chance” scheme
  – Simple multi-threading for easy HW implementation and efficient hiding of memory latency
  – Cache miss triggers the ray thread that is set to idle mode
  – Ray thread is set to active mode at the next loop to re-access the cache; a cache miss acts as pre-fetching data for the next loop

Two-level cache hierarchy
  – L1/L2 caches
  – L1/L2 Address FIFO for handling memory requests
  – L1/L2 Address/Data FIFO for delivering address & data to the upper-level cache
RayTree® – RayTree® Architecture

- Fast $kd$-tree building without tree-quality degradation
  - Binning method [3] for making upper-level nodes, called scan-tree
  - Sorting method [4] for making lower-level nodes, called $kd$-tree

- Minimized off-chip memory access
  - Internal SRAM in the sorting-based pipeline for sorting, split plane selection, and geometry classification without external DRAM accesses

- Exploitation of a burst memory access
  - Reallocates a node construction sequence as the depth-first layout in node scheduler


Performance
Performance – Test Bench for Ray-tracing Performance
Performance – Ray-tracing Performance

- Ray-tracing performance:
  - 100M rays/s (MRPS), 45FPS@720p resolution

<table>
<thead>
<tr>
<th>Test scene</th>
<th>No. of primitives</th>
<th>Display resolution</th>
<th>Graphic effects</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cup</td>
<td>5,356</td>
<td>HD (720p)</td>
<td>Ray-tracing effects*</td>
<td>45.70</td>
</tr>
<tr>
<td>Kitchen</td>
<td>189,202</td>
<td>HD (720p)</td>
<td>Ray-tracing effects*</td>
<td>14.26</td>
</tr>
<tr>
<td>Lake</td>
<td>9,195</td>
<td>HD (720p)</td>
<td>Ray-tracing effects*</td>
<td>23.56</td>
</tr>
<tr>
<td>Living room</td>
<td>287,213</td>
<td>HD (720p)</td>
<td>Ray-tracing effects*</td>
<td>16.07</td>
</tr>
<tr>
<td>Mobil</td>
<td>164,430</td>
<td>HD (720p)</td>
<td>Ray-tracing effects*</td>
<td>10.39</td>
</tr>
<tr>
<td>Orgel</td>
<td>6,338</td>
<td>HD (720p)</td>
<td>Ray-tracing effects*</td>
<td>18.48</td>
</tr>
<tr>
<td>Pebble UI</td>
<td>61,678</td>
<td>HD (720p)</td>
<td>Ray-tracing effects*</td>
<td>20.79</td>
</tr>
<tr>
<td>Watch</td>
<td>10,048</td>
<td>HD (720p)</td>
<td>Ray-tracing effects*</td>
<td>13.38</td>
</tr>
<tr>
<td>Waterdrop UI</td>
<td>20,982</td>
<td>HD (720p)</td>
<td>Ray-tracing effects*</td>
<td>17.81</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>20.05</strong></td>
</tr>
</tbody>
</table>

* Ray-tracing effects: Reflection, refraction, transmission, shadow; all in real-time
Performance – Test Bench for Accel. Structure Building Performance
### Performance – Acceleration Structure Building Performance

- Acceleration Structure building performance for dynamic scenes:
  - 1M triangles/sec

<table>
<thead>
<tr>
<th>Test scene</th>
<th>No. of primitives</th>
<th>Desktop PC*</th>
<th>Mobile AP**</th>
<th>RayTree® ***</th>
</tr>
</thead>
<tbody>
<tr>
<td>Church</td>
<td>972</td>
<td>2.0074</td>
<td>23.8047</td>
<td>0.6475</td>
</tr>
<tr>
<td>Kitchen</td>
<td>1,079</td>
<td>3.0599</td>
<td>38.1295</td>
<td>0.6266</td>
</tr>
<tr>
<td>Watch</td>
<td>1,938</td>
<td>5.9568</td>
<td>74.2274</td>
<td>1.2198</td>
</tr>
<tr>
<td>Vegetable_lens</td>
<td>2,720</td>
<td>7.7652</td>
<td>86.2227</td>
<td>1.6932</td>
</tr>
<tr>
<td>Lake</td>
<td>6,903</td>
<td>22.0243</td>
<td>244.5517</td>
<td>4.8024</td>
</tr>
<tr>
<td>Mobil</td>
<td>7,208</td>
<td>25.5788</td>
<td>242.6982</td>
<td>3.3539</td>
</tr>
<tr>
<td>Flight_simulation</td>
<td>10,856</td>
<td>35.1052</td>
<td>271.5869</td>
<td>7.6767</td>
</tr>
<tr>
<td>Vegetable_dynamic</td>
<td>30,762</td>
<td>51.6724</td>
<td>457.9666</td>
<td>19.5148</td>
</tr>
</tbody>
</table>

* Desktop PC: Intel i3-2120@3.3GHz, using single thread
** Mobile AP: ARM Cortex A15@1.7GHz, using single thread
*** RayTree®: one scan-tree unit and two kd-tree units@266MHz
Performance – Real-time Ray-tracing Demo

- **Lake:**
  - Number of primitives: 9,195
  - Number of light sources: 2
  - Number of ray bounces: 0~14

- **Ray-tracing effects:**
  - Reflection on dynamic lake surface, transmission on boat sail, simultaneous changes in reflection on lake surface due to change in background image
Performance – Real-time Ray-tracing Demo

- Living room:
  - Number of primitives: 287,213
  - Number of light sources: 2
  - Number of ray bounces: 0~14

- Ray-tracing effects:
  - Transmission on table, refraction on red cup, global shadow of each object, bump mapping on book on footstool

-Ray bounce of 0>
-Ray bounce of 14>
Performance – Real-time Ray-tracing Demo

- **Waterdrop UI:**
  - Number of primitives: 20,949
  - Number of light sources: 2
  - Number of ray bounces: 0~14

- **Ray-tracing effects:**
  - Refraction on dynamic waterdrop folders, simultaneous changes in refraction on waterdrop folders due to change in user-customized background image
Performance – Real-time Ray-tracing Demo

- **Mobil:**
  - Number of primitives: 194,430
  - Number of light sources: 3
  - Number of ray bounces: 0~14

- **Ray-tracing effect:**
  - Reflection on window and water surface on bowl, transmission on transparent table, global shadow on every object, simultaneous changes in global shadow due to moving light source
RayChip® is the world’s first commercialized chip targeted to realize real-time ray tracing for embedded applications such as TV, media box and game console.

RayChip® includes real-time ray-tracing HW unit, called RayCore®, and acceleration structure building HW unit, called RayTree®.

RayCore® has fully hardwired, pipelined architecture:
- MIMD processing of ray threads
- Scalable architecture based on tile scheduling
- Pipeline efficiency improvement using “looping for the next chance” scheme

RayTree® is fully hardwired acceleration structure building unit:
- Parallel hybrid tree building architecture
- One scan-tree unit & two kd-tree units

RayCore® API allows easy-to-use programming environment for ray-tracing.
Future Plan

- **Virtual Reality (‘VR’) Platform**
  - Ray-tracing and sound-tracing technologies are combined to provide more immersive VR experience
  - Sound-tracing HW IP based on ray-tracing will be released in the near future

- **RayChip® Series 2000 Chip**
  - Advanced graphics functions such as soft shadow, ambient occlusion, displacement mapping, etc. are to be added
  - Ray-tracing GPU and OpenGL ES 2.0/3.0 GPU is integrated to seamlessly deliver maximum graphic effects and to support existing 3D graphic contents

- **Real-time Global Illumination (‘GI’)**
  - Real-time GI, a photorealistic graphic algorithm, which contains indirect illumination model will be implemented based on path-tracing algorithm and noise filter techniques
Q&A