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

Woo-Chan Park[1,2], **Hee-Jin Shin[1]**, Byoungok Lee[1], Hyungmin Yoon[1], and Tack-Don Han[1,3]

[1]**Siliconarts, Inc.**, [2]Sejong University, [3]Yonsei University

heejinshin@siliconarts.com

Copyright 2014 Siliconarts, Inc. All rights reserved

Table of Contents

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

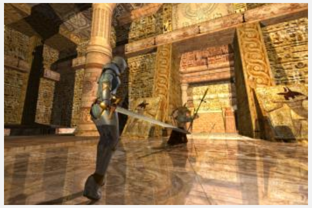
- Color impression on objects

OpenGL ES 2.0/3.0

- Modifiable object color, modifiable imagery pixel color

Global Illumination

Basic Ray-tracing



- Major effect of light – reflection, refraction, transmission, and shadow

RayChip® Series 1000

Advanced Ray-tracing

- Realistic shadow and curvature by calculating volume of light

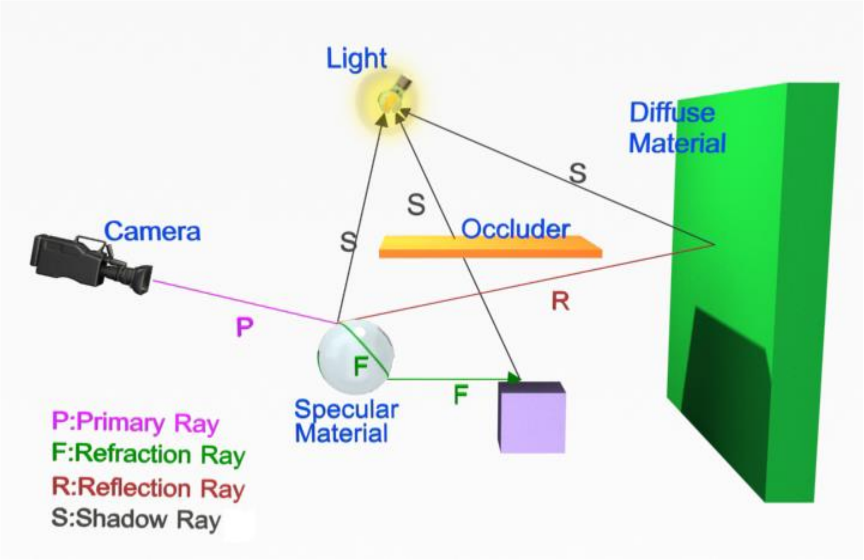
Distribution Ray-tracing & Indirect Illumination*

- 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



<Ray-tracing Algorithm>

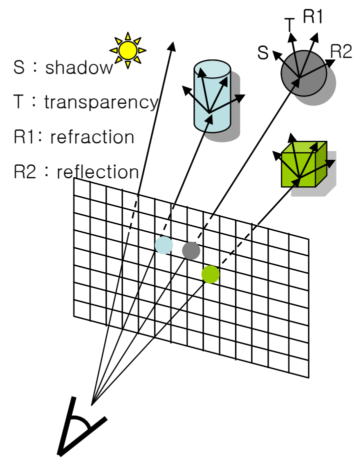
- **Advantages of Ray-tracing algorithm [1]**
  - Supporting global illumination effects such as reflection, refraction, shadow, transmission
  - Less computational complexity in object numbers (e.g., O (log N))
  - On-demand computation
  - Declarative scene description
  - Parallel (as nature)

- **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

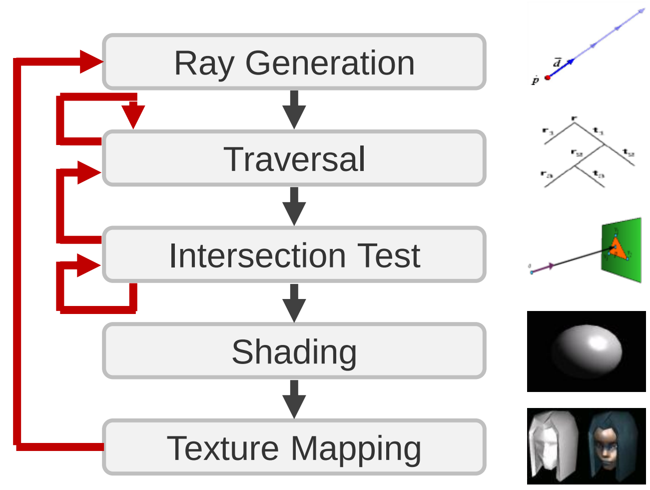
* [1] Jim Hurley, “Ray Tracing Goes Mainstream,” *Intel Technology Journal*, Vol. 9, pp. 98-107, 2005.

Ray-tracing Algorithm – Ray-tracing VS. Rasterization

Ray-tracing

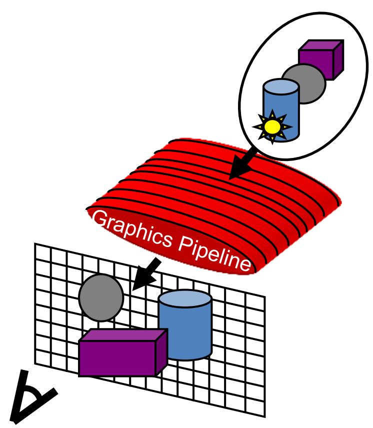


<Ray-tracing Fundamental>

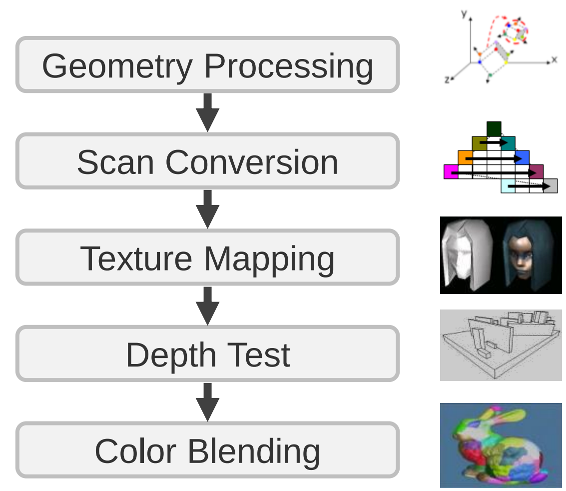


<Ray-tracing GPU Pipeline>

Rasterization



<Rasterization Fundamental>

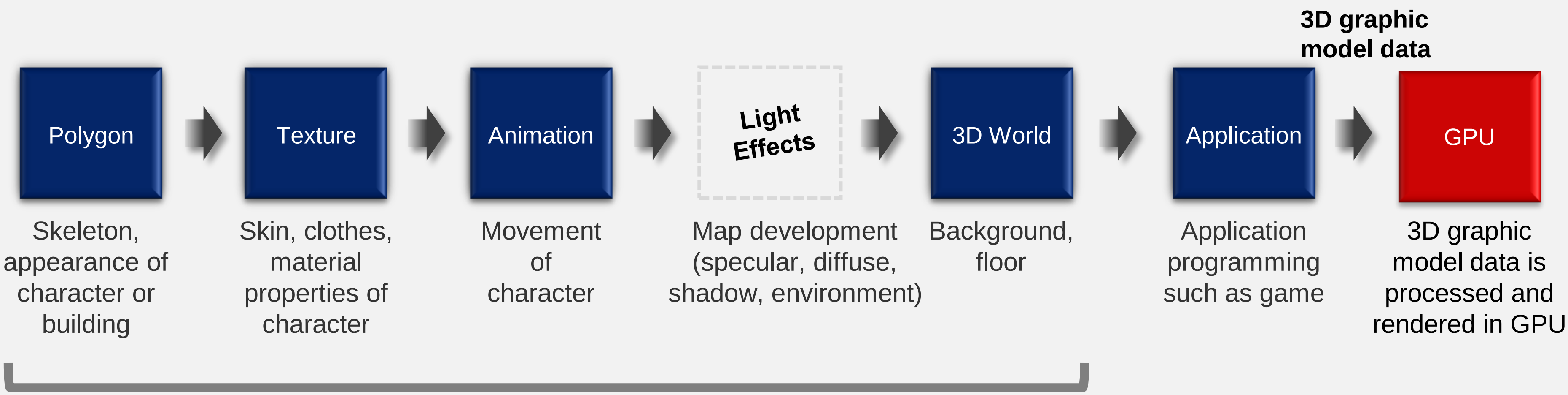


<Rasterization GPU Pipeline>

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

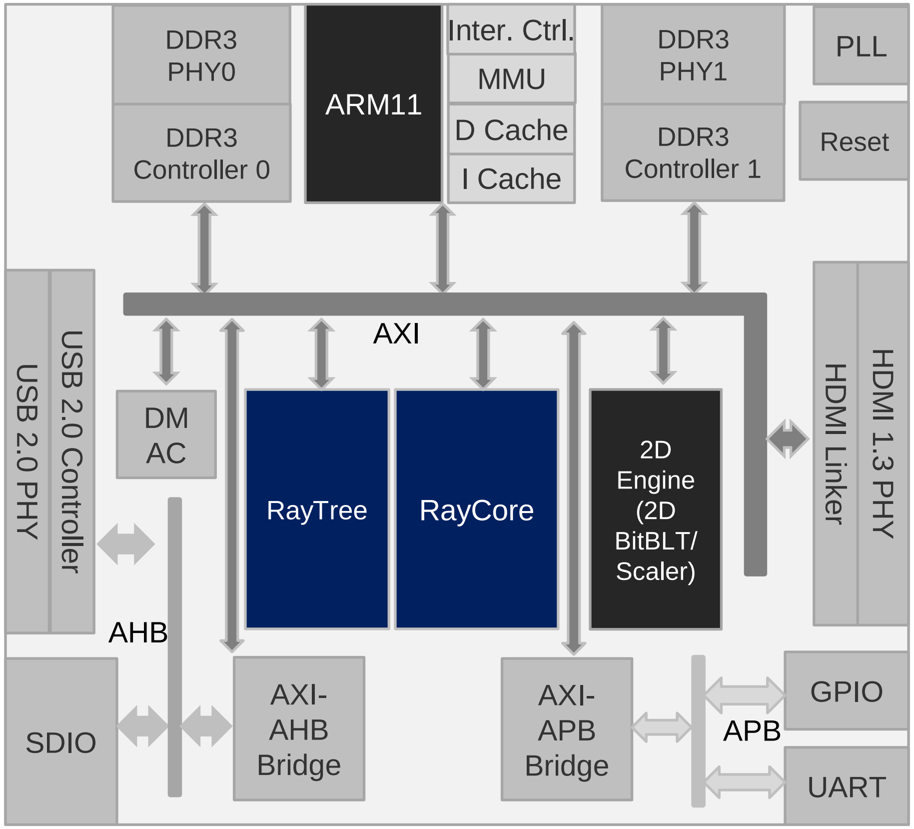
- The world's first real-time ray-tracing chip for embedded applications
  - Full hard-wired logic to achieve real-time performance for ray-tracing rendering
  - Processes multiple ray bounces recursively to create realistic images
  - Maintains high throughput pipeline by adopting MIMD parallel architecture 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

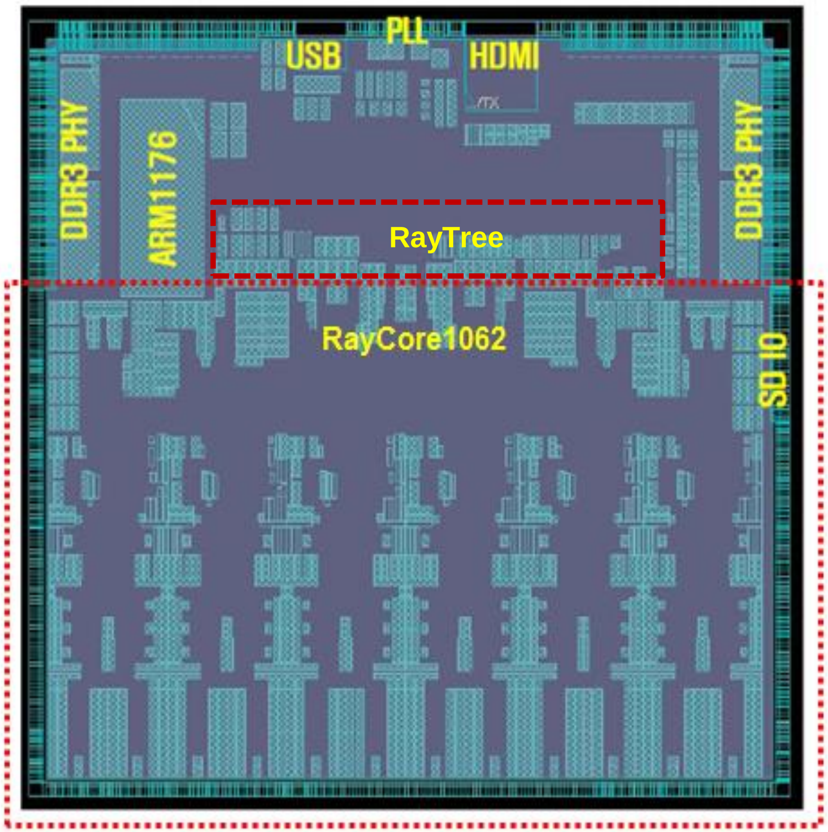
- Easy-to-use OpenGL ES-familiar API support
  - Provides a diverse set of graphics functionalities and an OpenGL ES 1.1-familiar API
  - Allows developers to create high-quality 3D graphics applications at lower cost

RayChip® Series 1000 – Overview

- 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

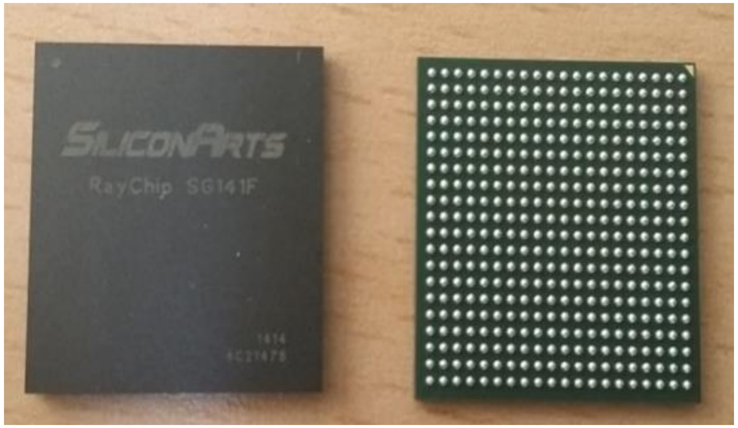


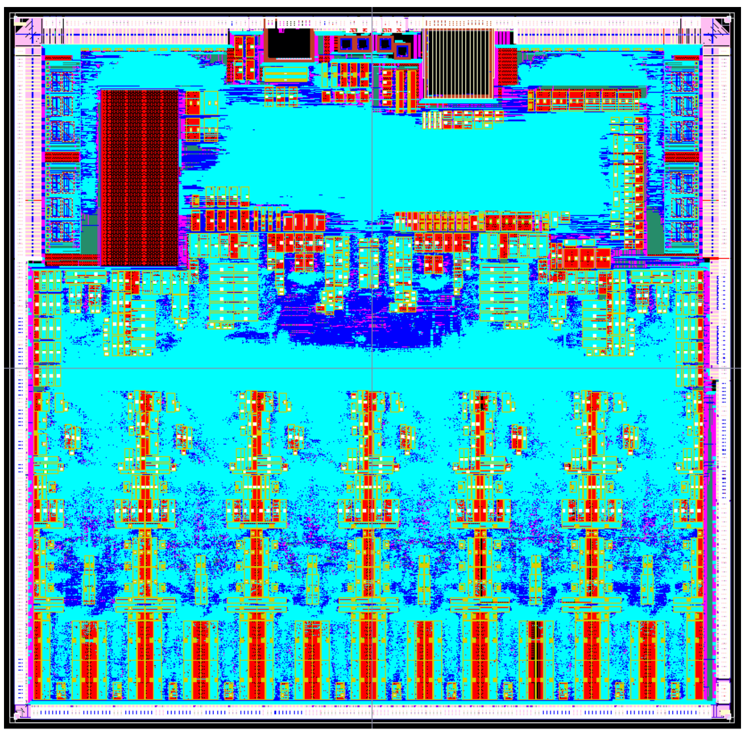
<RayChip® SF141F Block Diagram>



<RayChip® SF141F Floorplan>

RayChip® Series 1000 – Specifications

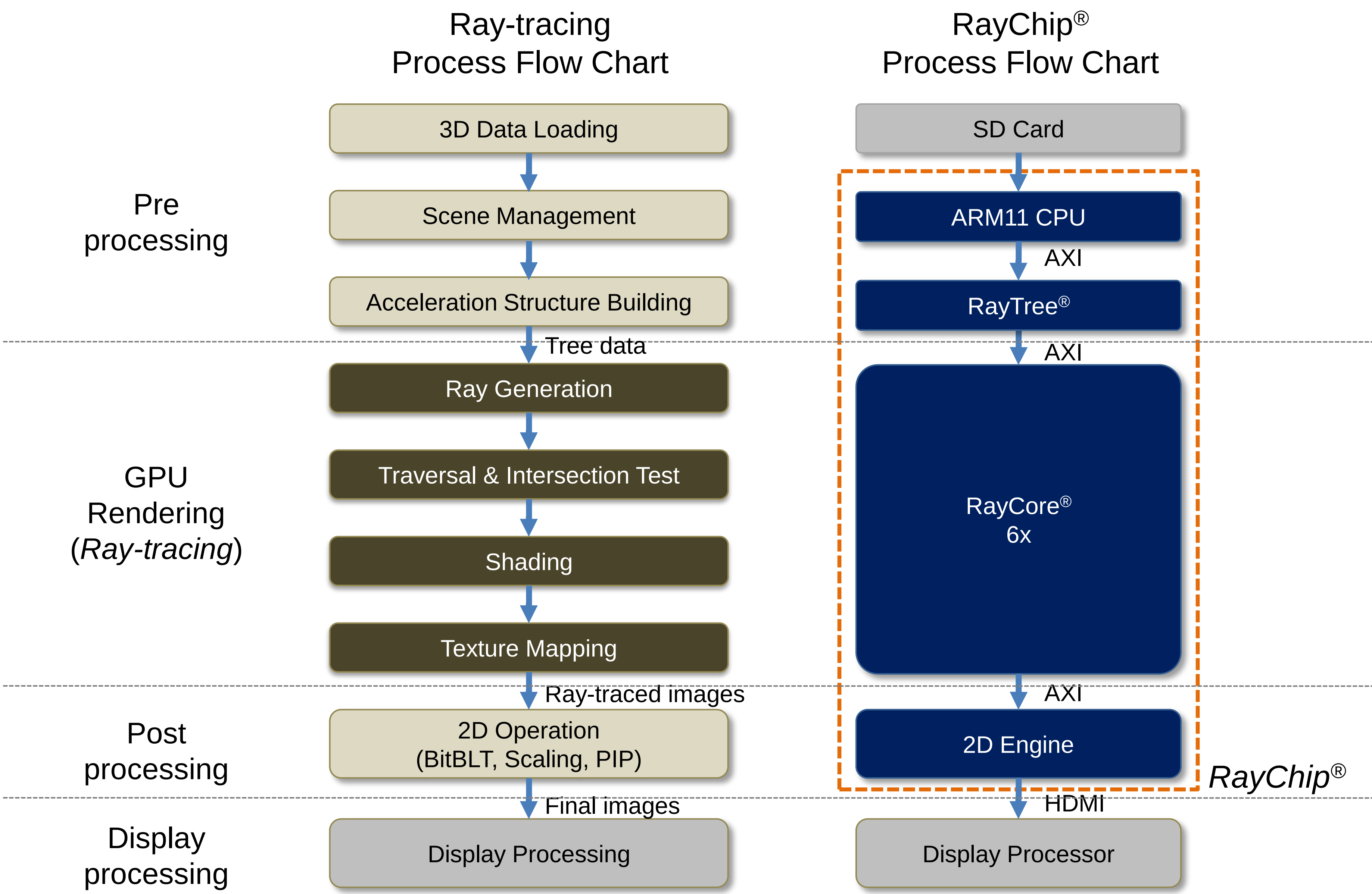




<RayChip® SG141F Die Shot>

Item	Description
Part No.	SG141F
Technology	Fujitsu 55nm low-power technology
Die area	9.6×9.4 mm^2
Package	17×17 mm, 400 FBGA
Voltage	Core 1.2V, I/O 1.8V, 3.3V
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

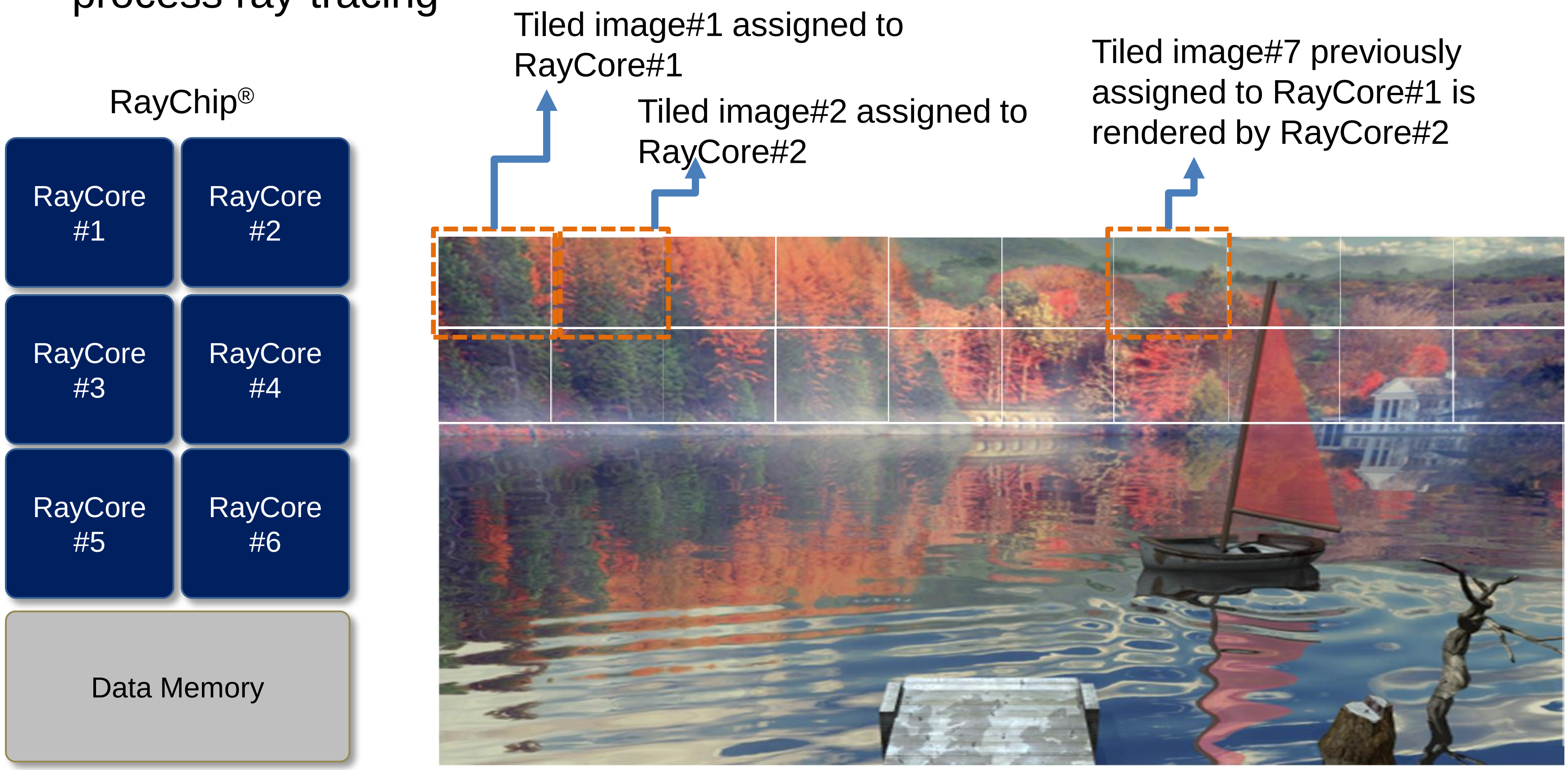
RayChip® Series 1000 – Process Flow



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® Series 1000 – RayCore® API

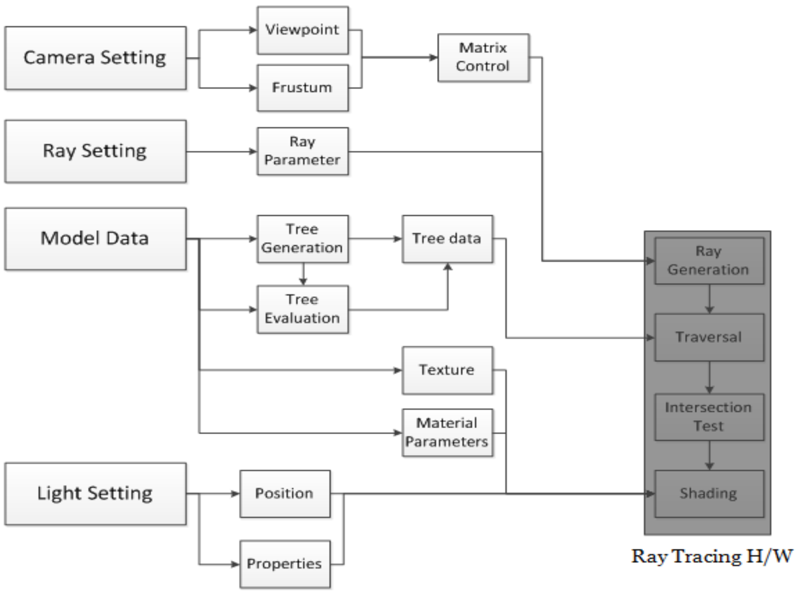
- Easy-to-use API for ray-tracing content development
  - Supports interface to develop ray-tracing 3D contents
  - Consists of API libraries similar to OpenGL ES 1.1 with ray-tracing specific functions

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

```
// 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>



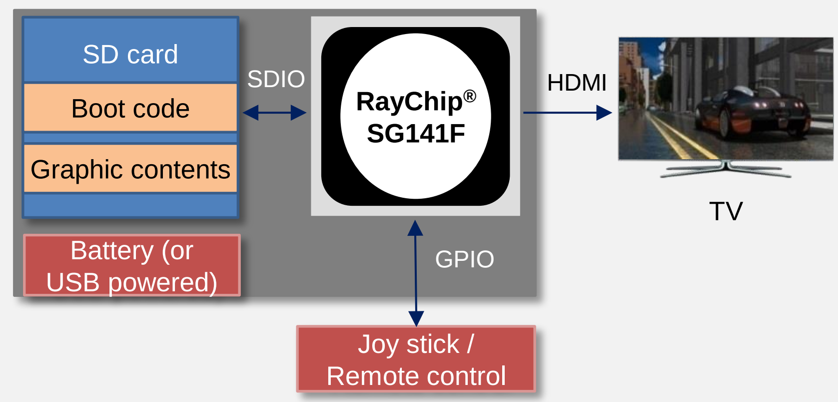
<RayCore® API Flow Chart>

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
Consumer TV	▪ Low power, dynamic 3D UI ▪ Intuitive, easy-to-use UX
Commercial TV / Digital Signage	▪ Virtual advertisement ▪ Augmented reality (AR)
Media Box / Dongle Mini PC	▪ Edutainment content with intuitive 3D UI and game
Game Console / Arcade Machine	▪ Immersive 3D game ▪ Reality-like virtual reality (VR)

Application Example



"Edutainment TV Dongle"

- Natural expression of high-quality 3D graphic content
- Maximizes user concentration
- Improves learning ability
- Interactive and easy-to-control UX

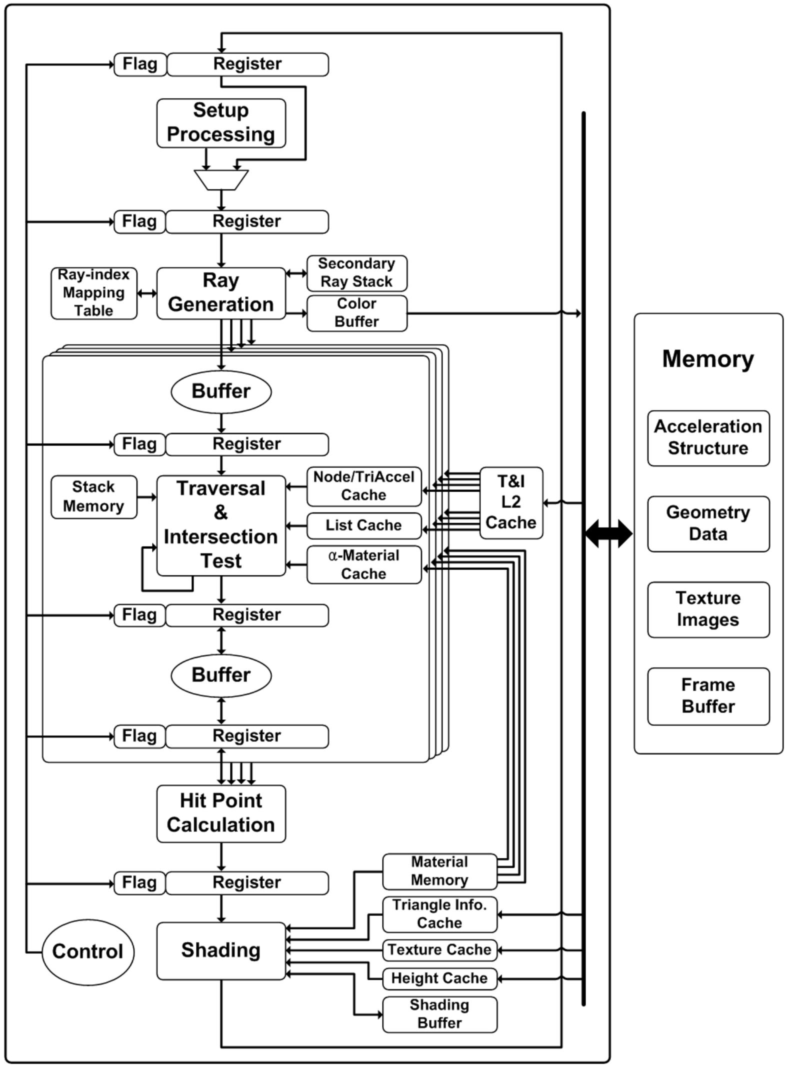
RayCore® and RayTree®

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



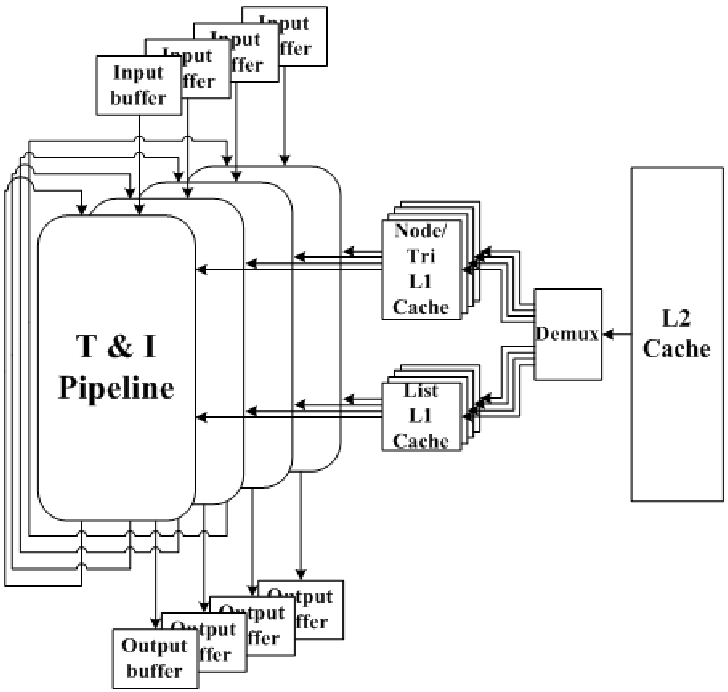
<RayCore® Architecture>

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]**

Category	SIMD	MIMD
Pros	Memory bandwidth reduction	Ideally perfect utilization
Cons	Low utilization rate in case of incoherent rays	Expensive HW cost
Performance	Good only in the case of coherent rays	Best (with L2 cache)

<MIMD vs. SIMD Pros and Cons>



<Unified T&I Pipeline>

TABLE V

COMPARING OUR PERFORMANCE ON TWO DIFFERENT CORE CONFIGURATIONS TO THE GTX285 FOR THREE BENCHMARK SCENES [11]. PRIMARY RAY TESTS CONSISTED OF 1 PRIMARY AND 1 SHADOW RAY PER PIXEL. DIFFUSE RAY TESTS CONSISTED OF 1 PRIMARY AND 32 SECONDARY GLOBAL ILLUMINATION RAYS PER PIXEL.

		Conference (282k triangles)		Fairy (174k triangles)		Sibenik (80k triangles)	
MIMD	Ray Type	MIMD Issue Rate	MIMD MRPS	MIMD Issue Rate	MIMD MRPS	MIMD Issue Rate	MIMD MRPS
$147mm^2$	Primary	74%	376	70%	369	76%	274
	Diffuse	53%	286	57%	330	37%	107
$175mm^2$	Primary	77%	387	73%	421	79%	285
	Diffuse	67%	355	70%	402	46%	131
SIMD	Ray Type	GTX SIMD eff.	GTX MRPS	GTX SIMD eff.	GTX MRPS	GTX SIMD eff.	GTX MRPS
GTX285	Primary	74%	142	76%	75	77%	117
	Diffuse	46%	61	46%	41	49%	47

MIMD MRPS/mm^2 ranges from 2.56 (Conference, primary rays) to 0.73 (Sibenik, diffuse rays) for both configs
SIMD MRPS/mm^2 ranges from 0.25 (Conference, primary rays) to 0.07 (Fairy, diffuse rays)
SIMD (no texture area) MRPS/mm^2 ranges from 0.47 (Conference, primary) to 0.14 (Fairy, diffuse)

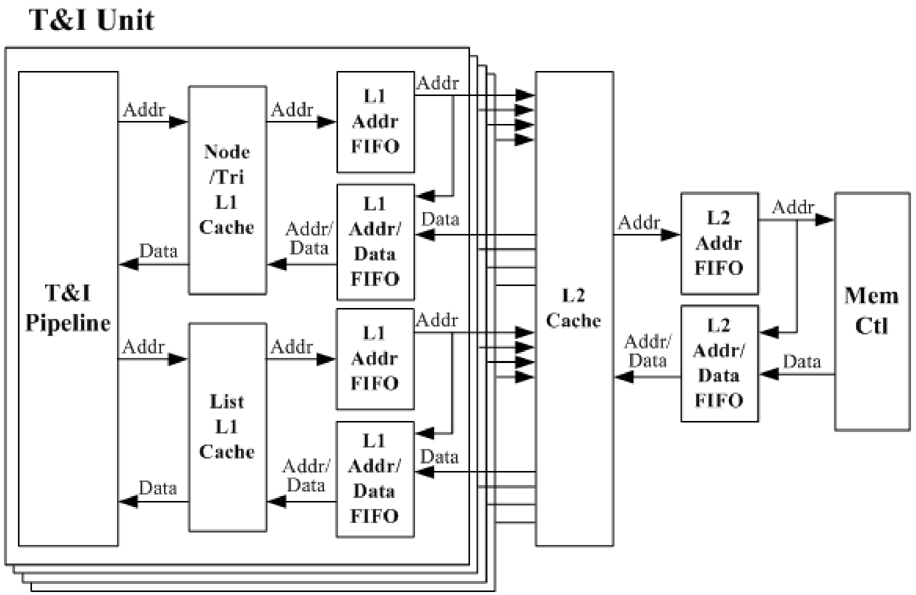
<MIMD vs. SIMD Performance Comparison>

* [2] D. Kopta, et al., "Efficient MIMD Architectures for High-Performance Ray Tracing," *IEEE International Conference on Computer Design*, pp. 9-16, Oct. 2010.

RayCore® – Memory Hierarchy

- "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

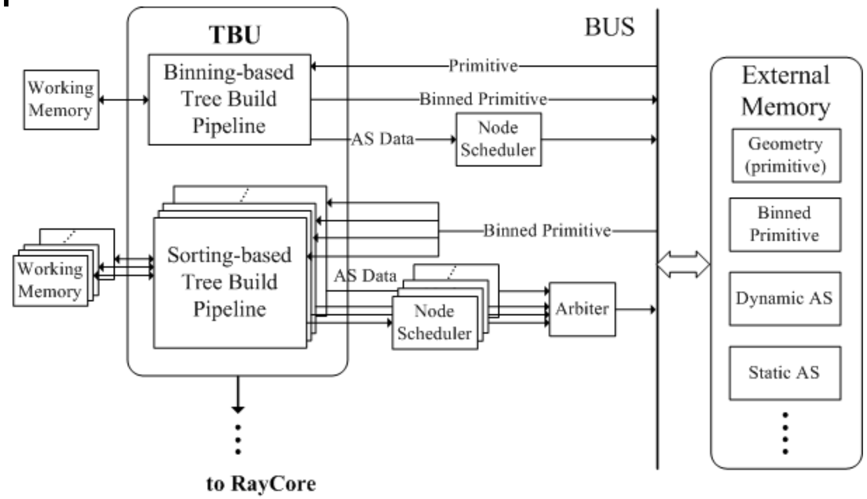
- Two-level cache hierarchy
  - L1/L2 caches
  - L1/L2 Address FIFO for handling memory requests
  - L1/L2 Address/Data FIFO for delivering address & data to the upper-level cache



<L1/L2 Memory Hierarchy>

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



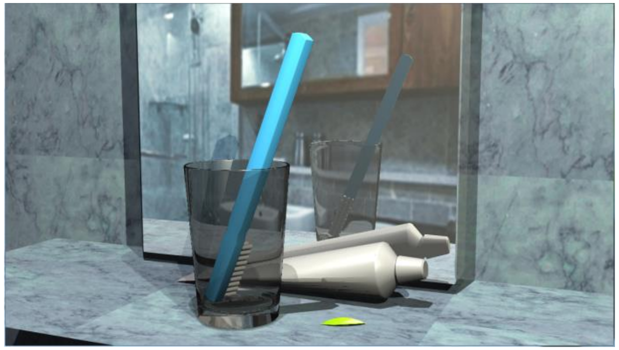
<RayTree® Architecture>

* [3] M. Shevtsov, A. Soupikov, and A. Kapustin, "Highly parallel fast *kd*-tree construction for interactive ray tracing of dynamic scenes," *Computer Graphics Forum*, Vol. 26, No. 3, pp. 395-404, Sept. 2007.

* [4] I. Wald and V. Havran, "On Building Fast *kd*-trees for Ray Tracing, and on Doing That in O(N log N)," *IEEE Symposium on Interactive Ray Tracing*, pp. 61-69, 2006.

Performance

Performance – Test Bench for Ray-tracing Performance


<Cup>


<Kitchen>


<Lake>


<Living room>


<Mobil>


<Orgel>


<Pebble UI>

<Watch>

<Waterdrop UI>

Performance – Ray-tracing Performance

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

[unit: MRPS; FPS]

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

* Ray-tracing effects : Reflection, refraction, transmission, shadow; all in real-time

Performance – Test Bench for Accel. Structure Building Performance

<Church>

<Kitchen>

<Watch>

<Vegetable_lens>

<Lake>

<Mobil>

<Flight_simulation>

<Vegetable_dynamic>

Performance – Acceleration Structure Building Performance

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

[unit: ms]

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

* Desktop PC: Intel i3-2120@3.3GHz, using single thread

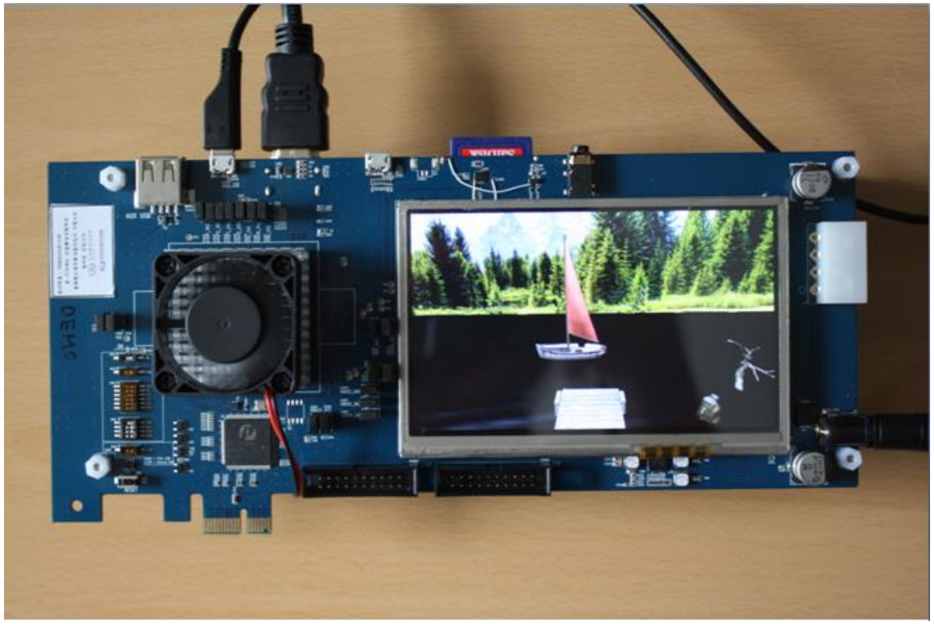
** Mobile AP: ARM Cortex A15@1.7GHz, using single thread

*** RayTree®: one scan-tree unit and two *kd*-tree units@266MHz

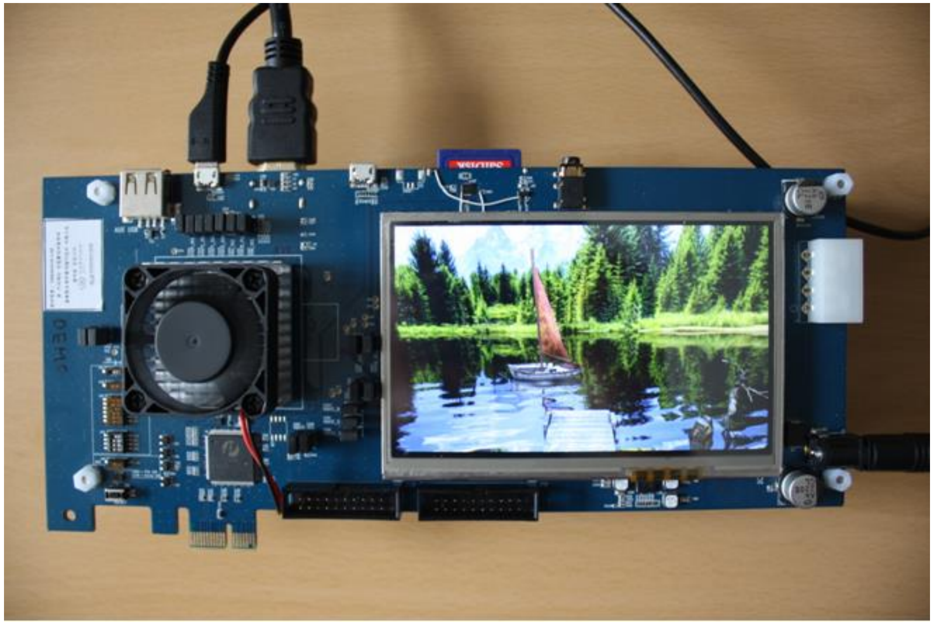
Performance – Real-time Ray-tracing Demo

- 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



<Ray bounce of 0>

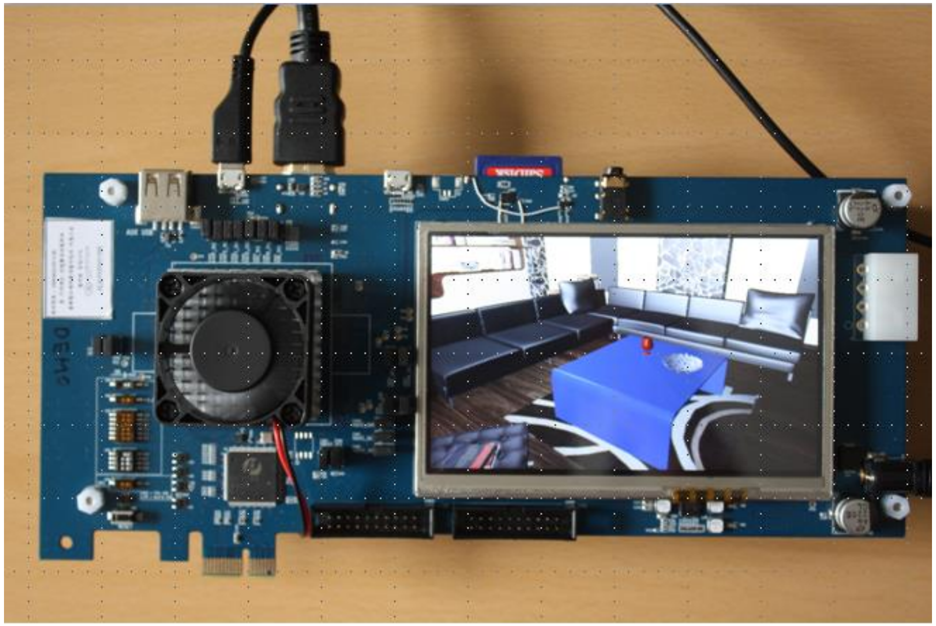


<Ray bounce of 14>

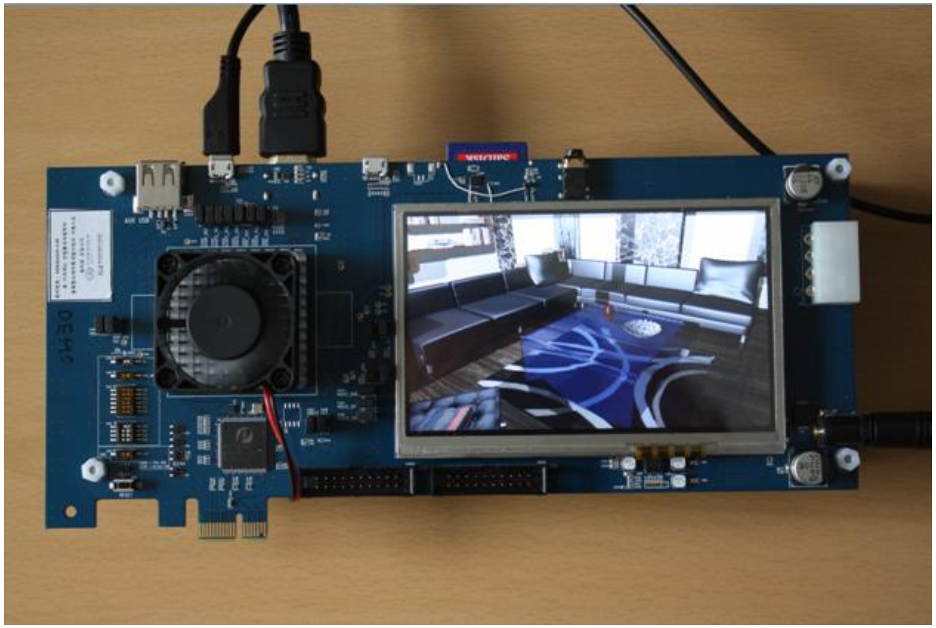
Performance – Real-time Ray-tracing Demo

- 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>

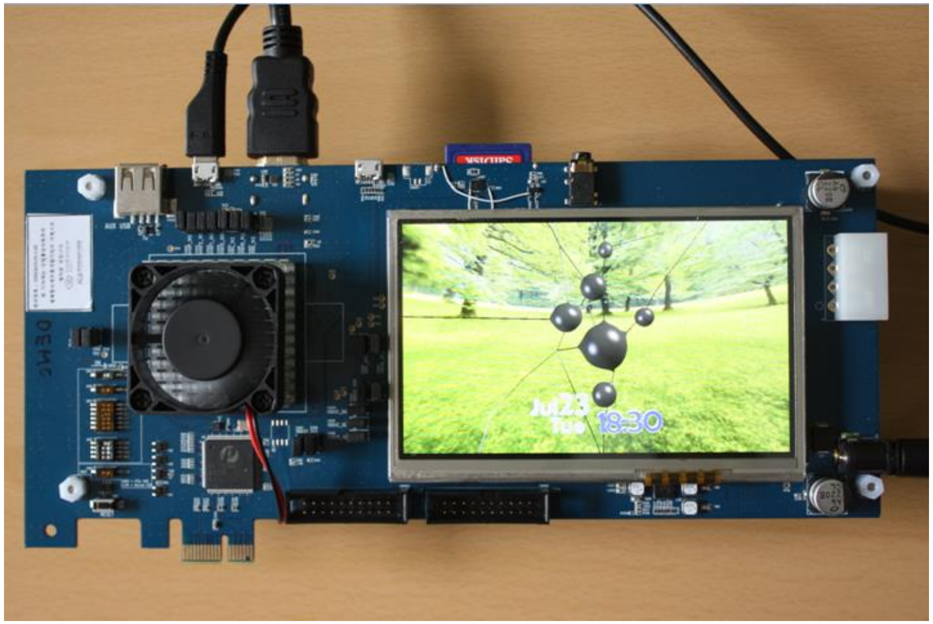


<Ray bounce of 14>

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



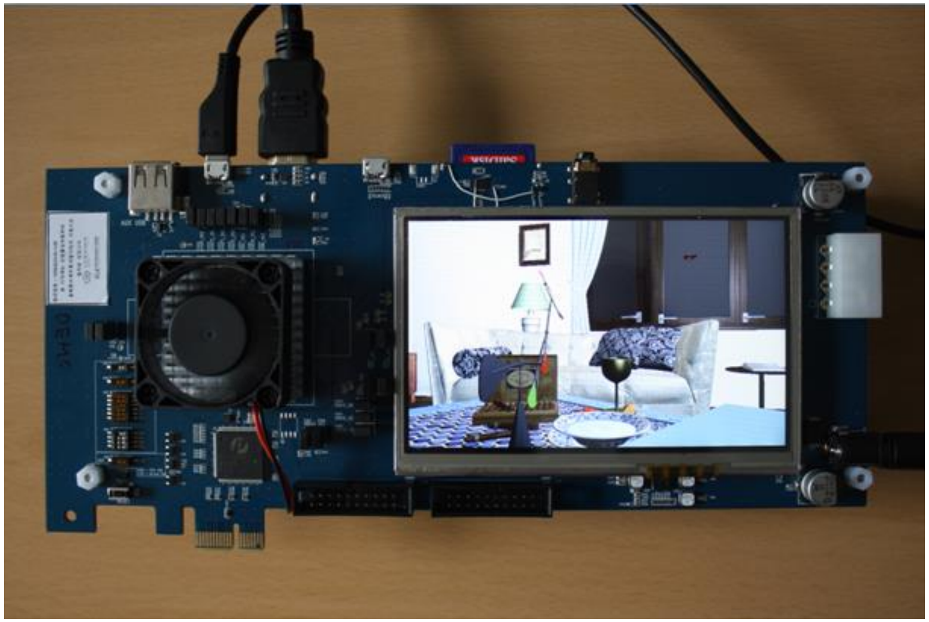
<Ray bounce of 0>

<Ray bounce of 14>

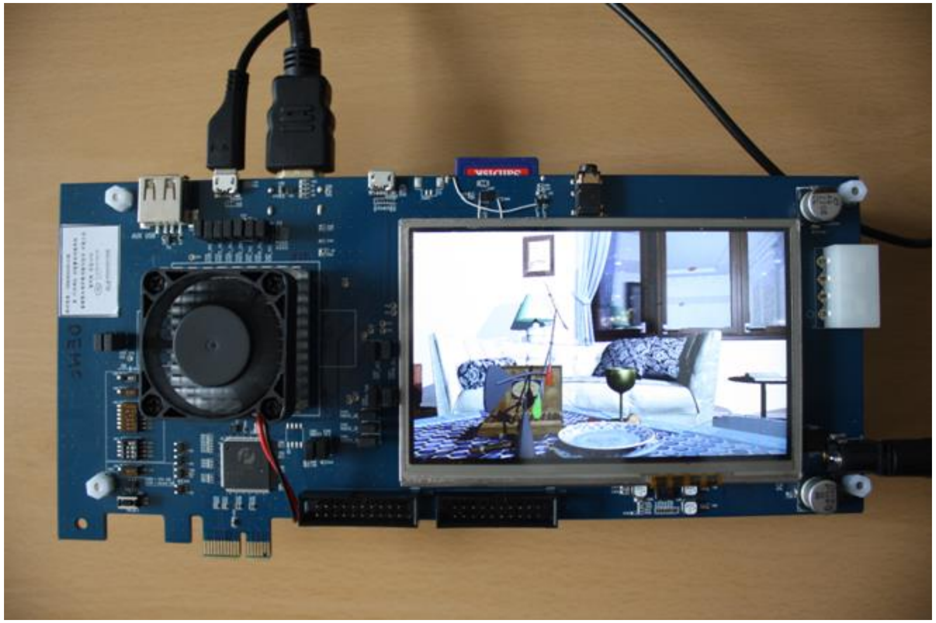
Performance – Real-time Ray-tracing Demo

- 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



<Ray bounce of 0>



<Ray bounce of 14>

Summary

- 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