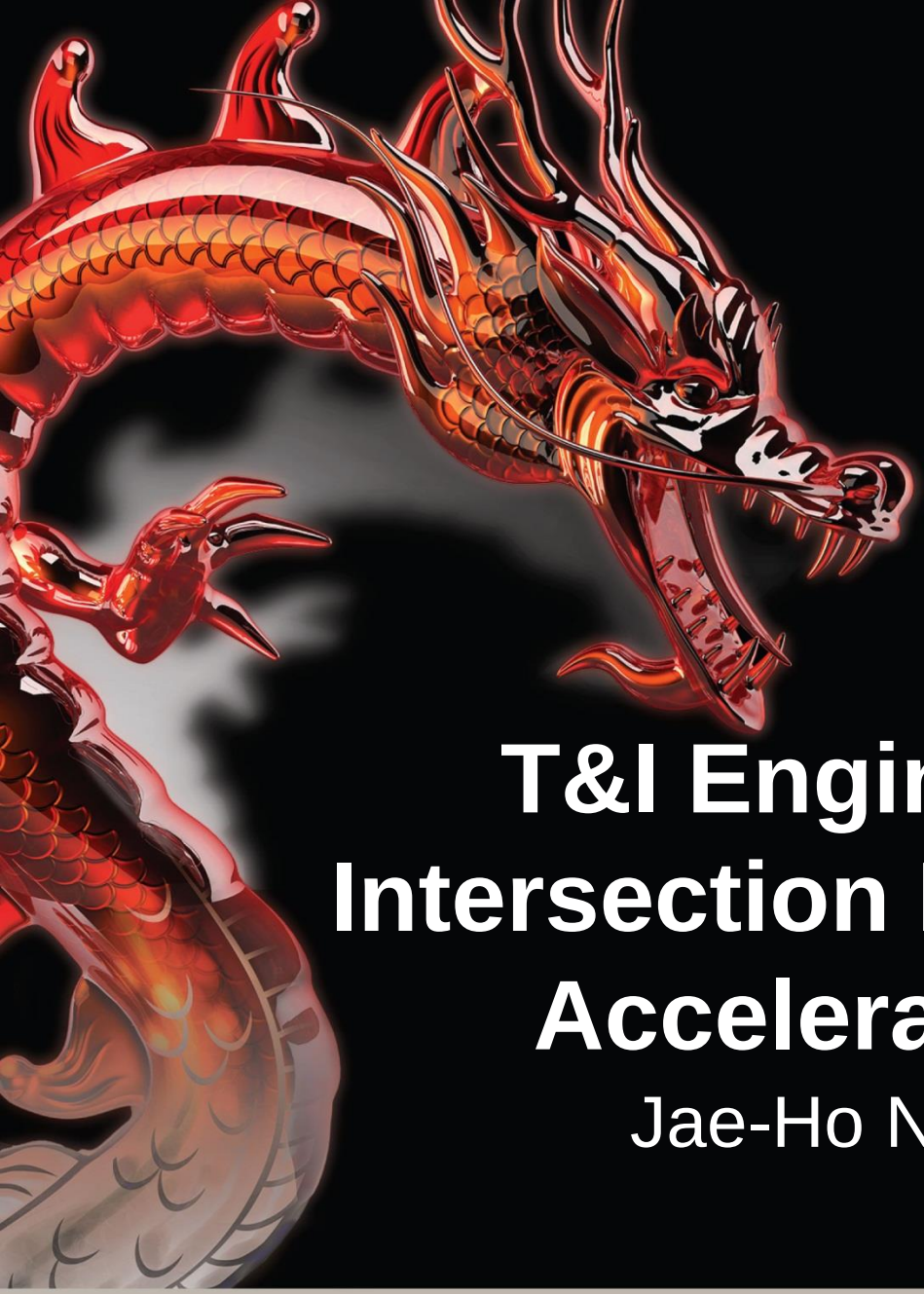
Sponsored by ACM SIGGRAPH

www.SIGGRAPH.org/ASIA2011



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

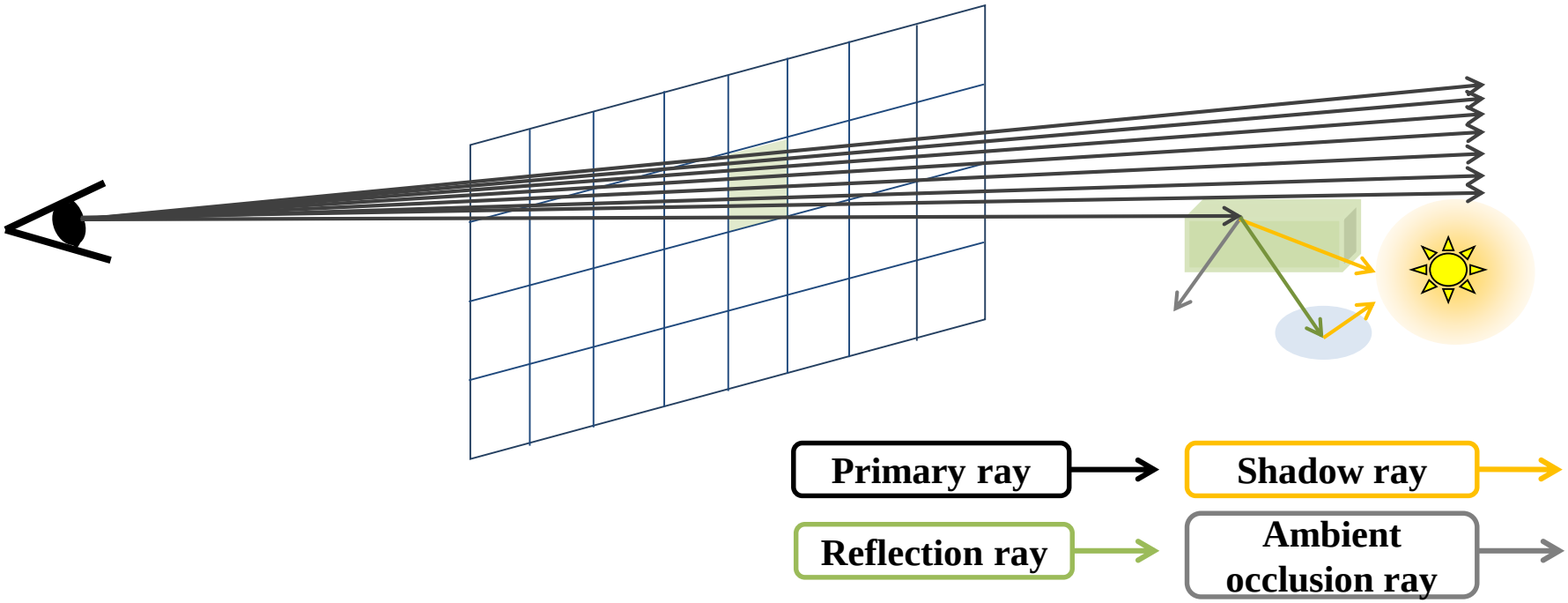
- Rendering technique by tracing the path of light
- Generates high-quality visual effects
  - Reflection, refraction, shadows, etc.
  - Widely used for off-line rendering
- Ray tracing goes mainstream for real-time rendering [Hurley 2005; Mark 2008]

← A ray-traced image created with POV-ray

Real-time Ray Tracing

- Performance requirement [Govindaraju et al. 2008]

: 1G rays/s ≒ $\underbrace{1024 \times 1024}_{\text{resolution}} \times \underbrace{8}_{\substack{\text{primary rays} \\ \text{per pixel}}} \times \underbrace{4}_{\substack{\text{secondary rays} \\ \text{per pixel}}} \times \underbrace{25}_{\substack{\text{frames per} \\ \text{second}}}$



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

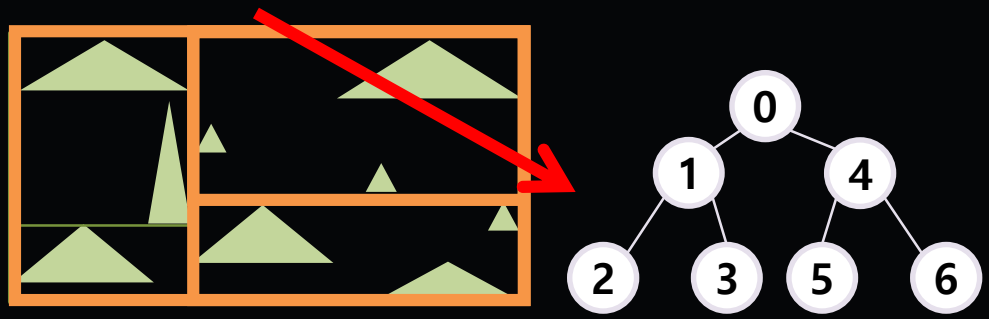
- General purpose many-core architecture
  - Wide-SIMD
    - Larrabee [Seiler et al. 2008] → Intel MIC
  - MIMD
    - Copernicus [Govindaraju et al. 2008]
    - xPU [Mahesri et al. 2008]
  - Fermi GPU-based architecture [Aila and Karras 2010]
- These approaches do not yet provide sufficient performance 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

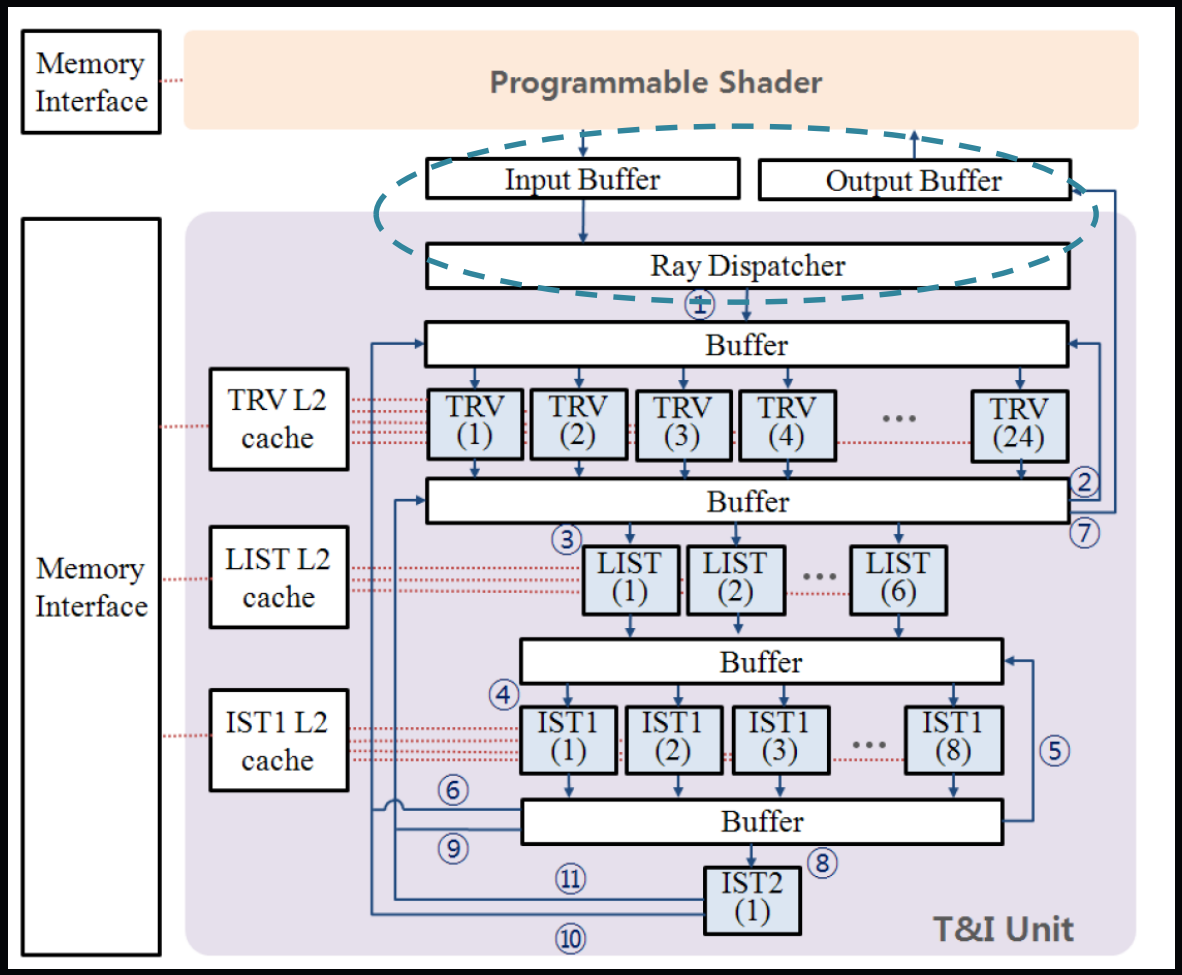
T&I Engine

- 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

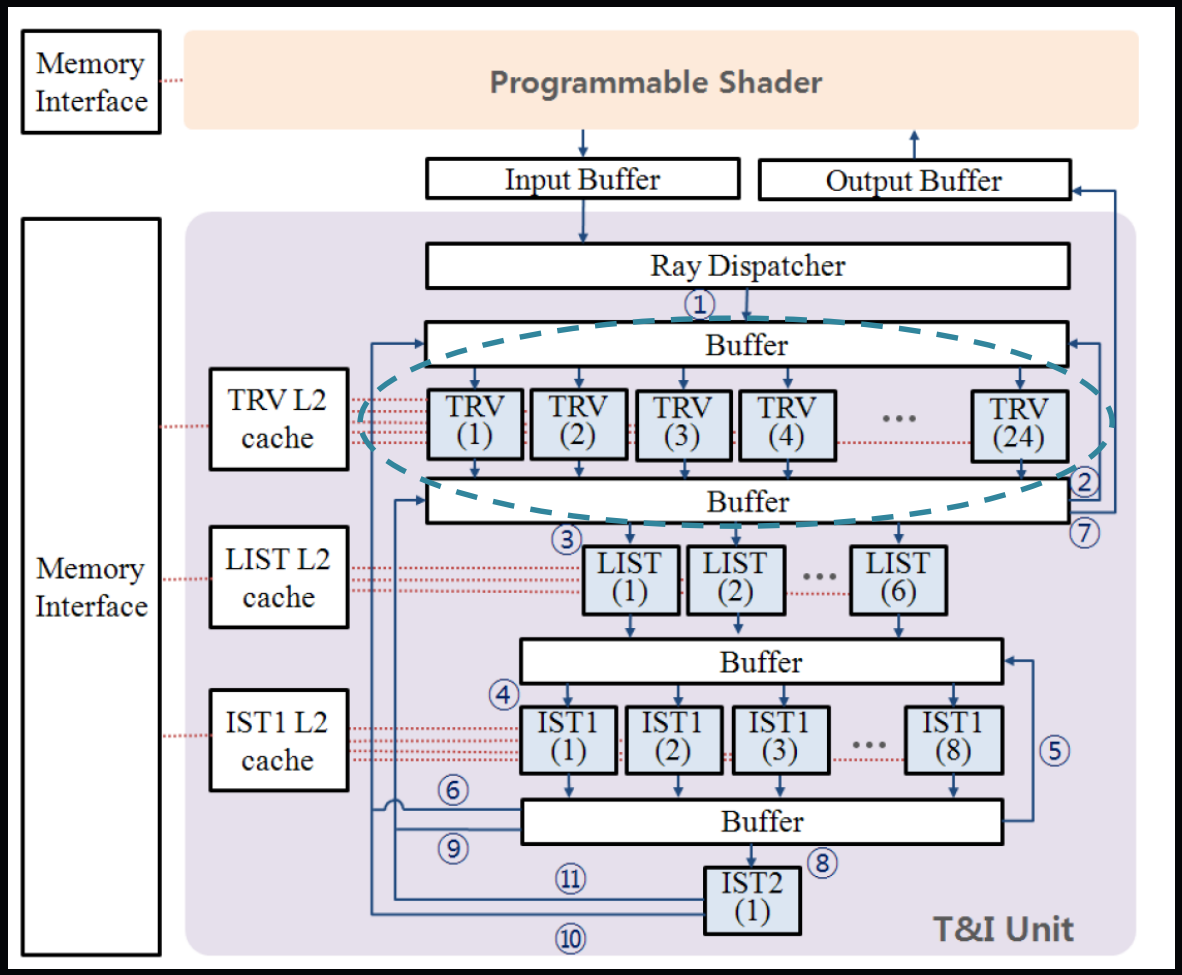
- Fixed logic design for T&I
  - High performance per area
  - Fully pipelined architecture
  - Programmable
    ray generation & shading
- Single ray tracing
  - Robust for incoherent rays
- Acceleration structure
  - *k*d-tree: best choice for
    single ray tracing

Overall System Architecture



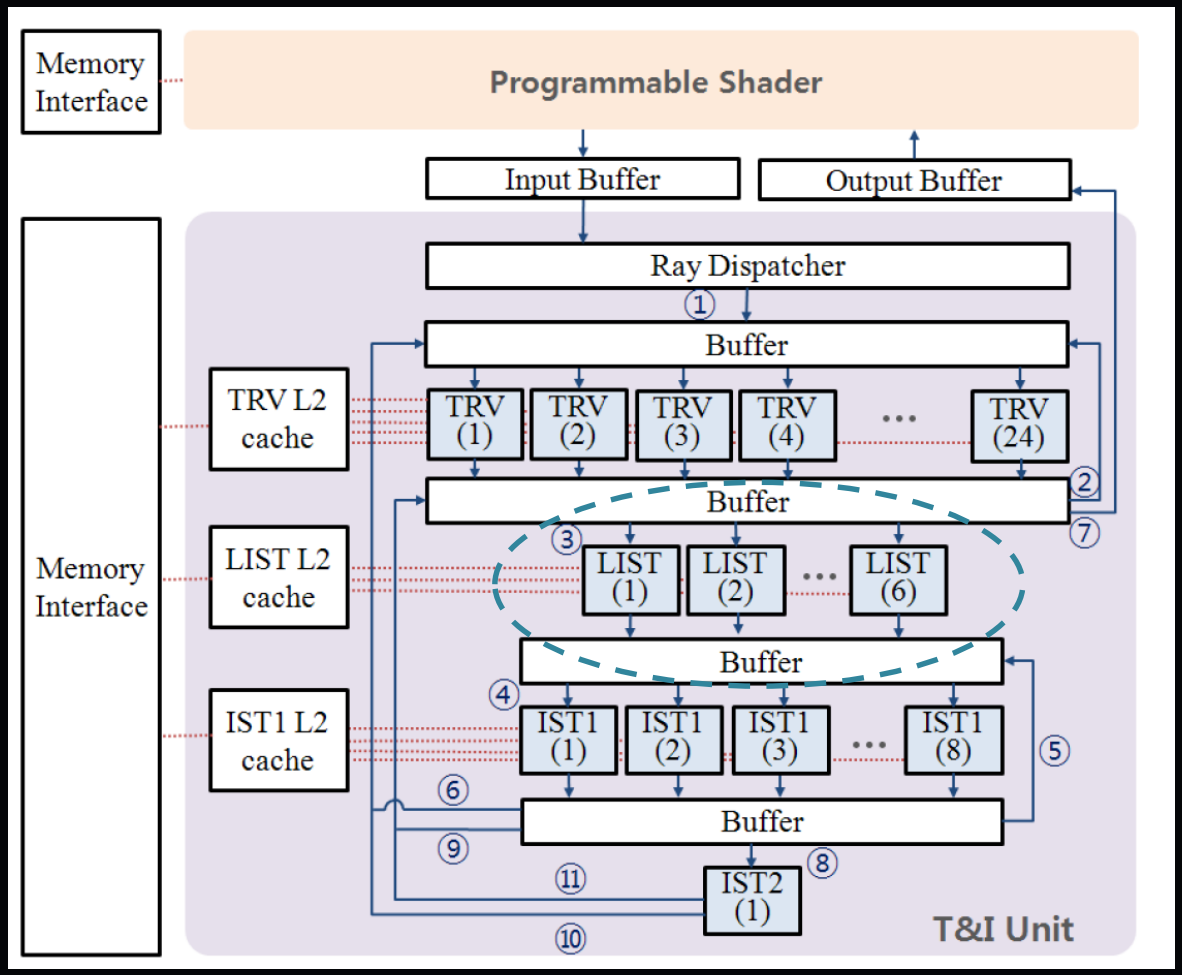
- Input / output buffers
- Ray dispatcher

Overall System Architecture



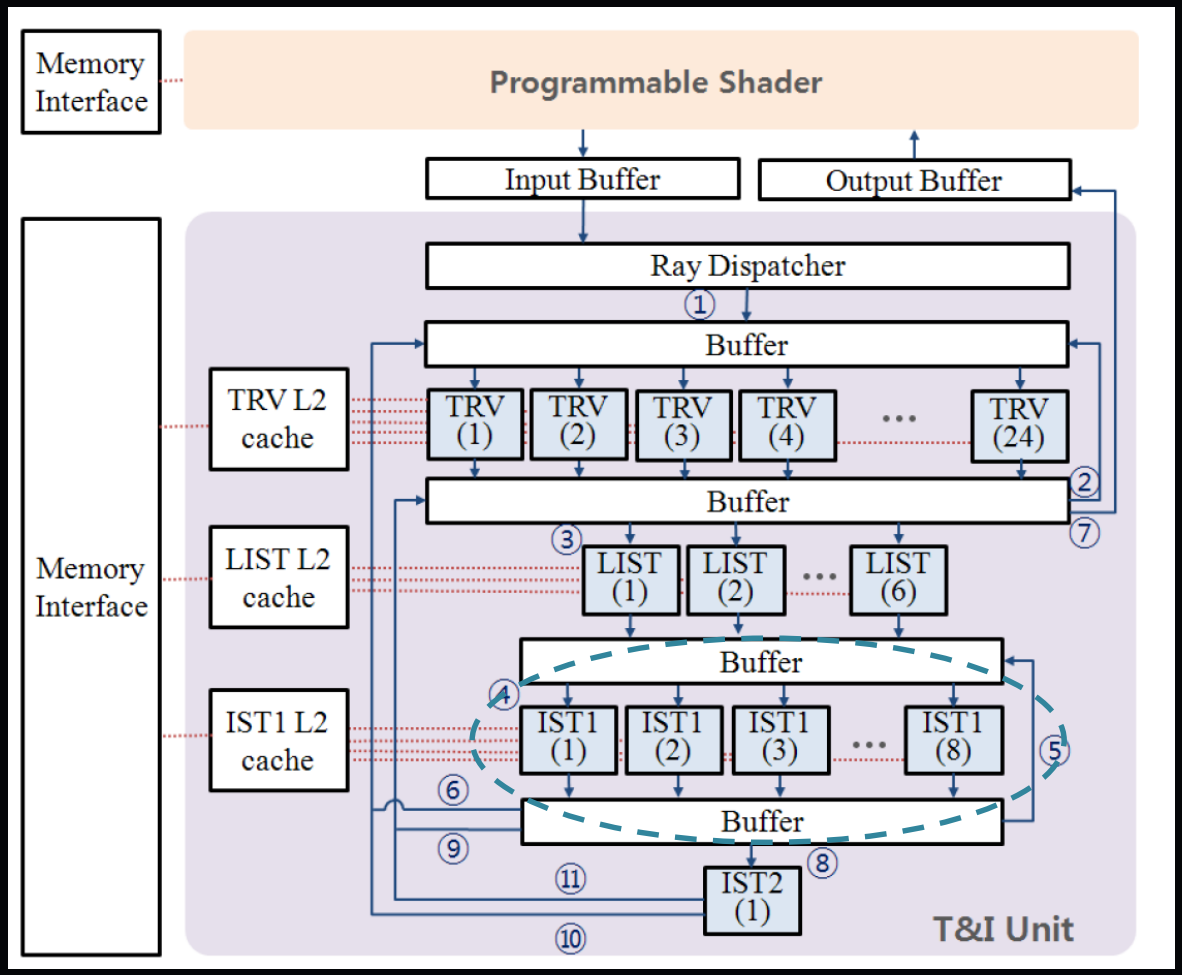
- Traversal units (TRVs)
  - *k*d-tree traversal

Overall System Architecture



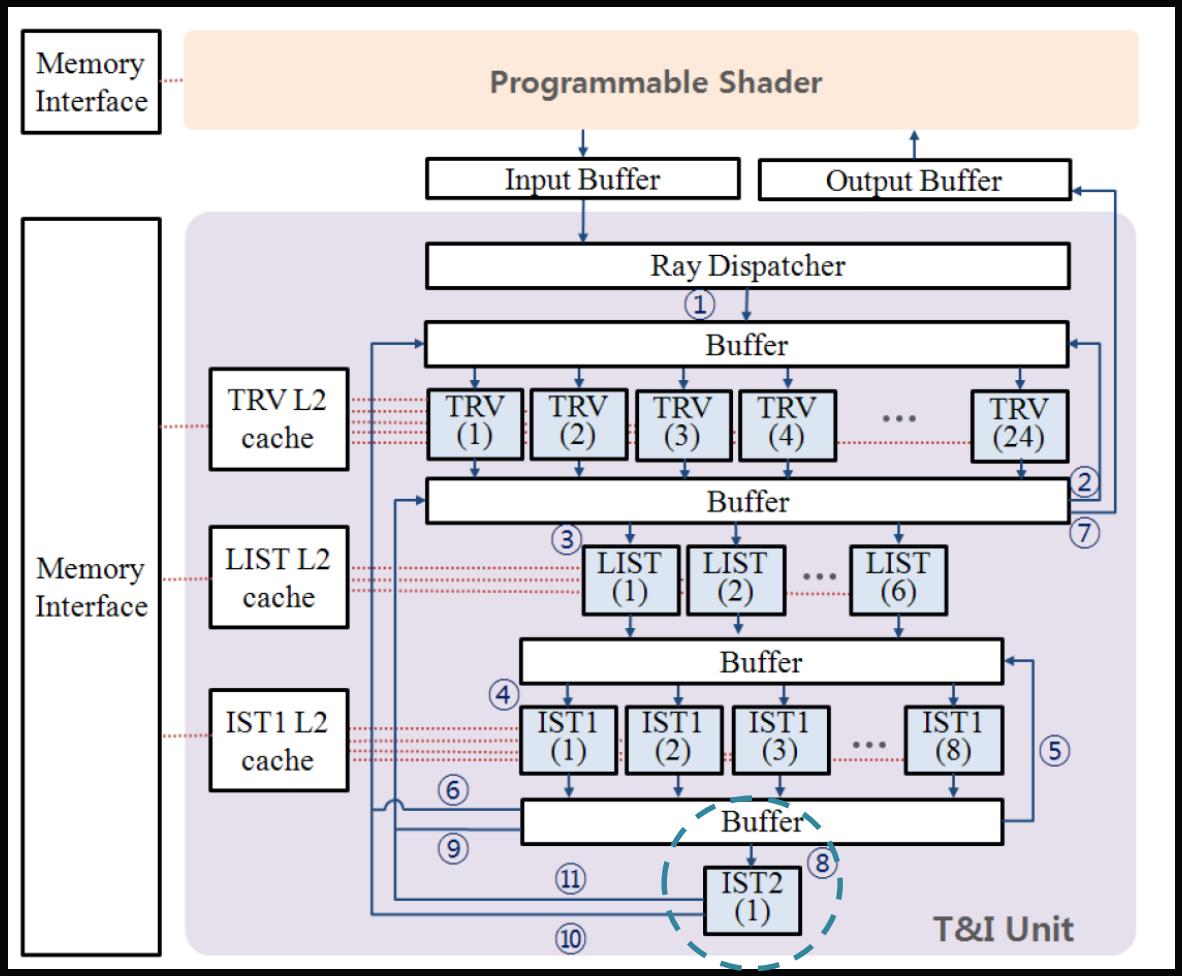
- List units (LISTs)
  - Search the primitive list in a leaf node

Overall System Architecture



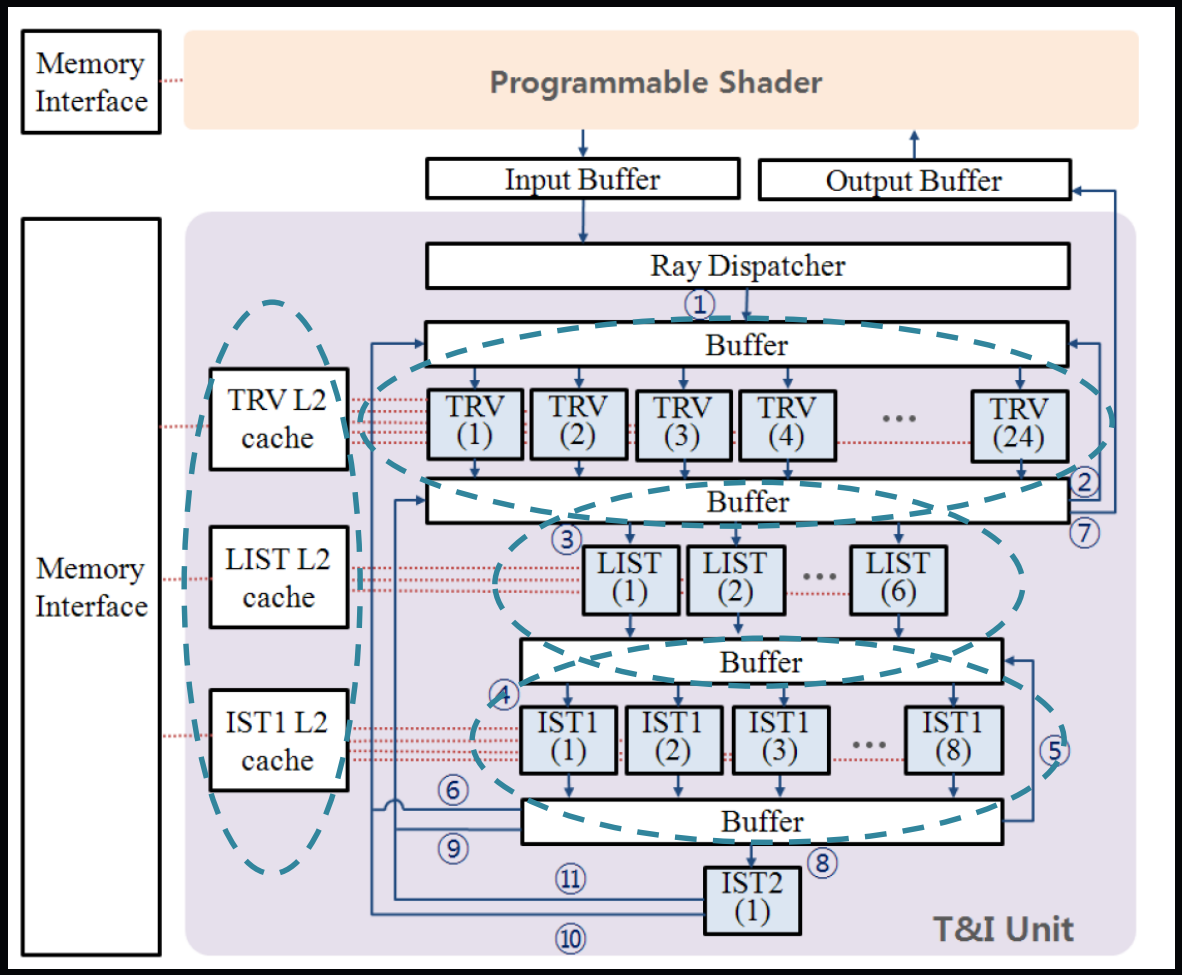
- The first intersection units (IST1s)
  - Ray-plane test
  - Barycentric test

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

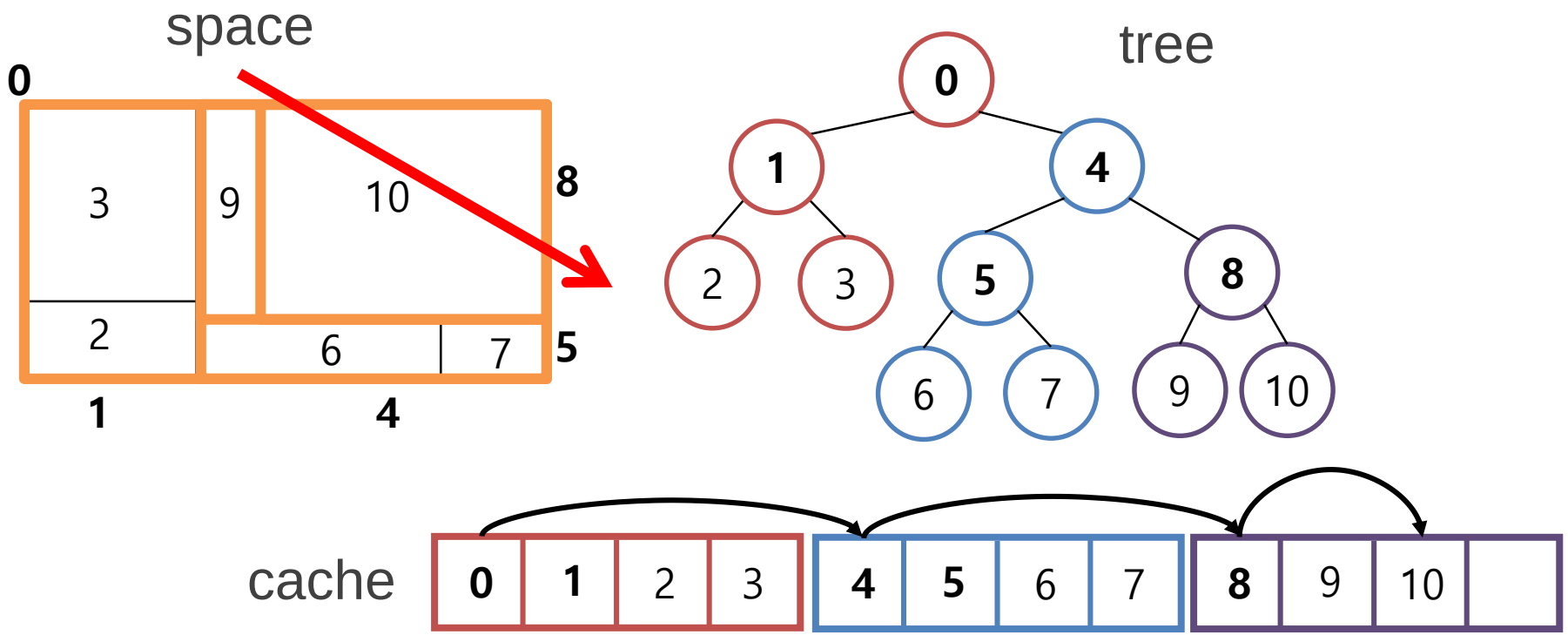
- We apply two methods to our architecture:
  - Ordered depth-first layout [Nah et al. 2010] (previously announced at SIGGRAPH ASIA Sketches)
  - Short-stack [Horn et al. 2007]
    - A small, *n*-entry stack to maintain the last *n* pushes
    - Reduces the required SRAM size for stacks

Introduction to ODFL

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

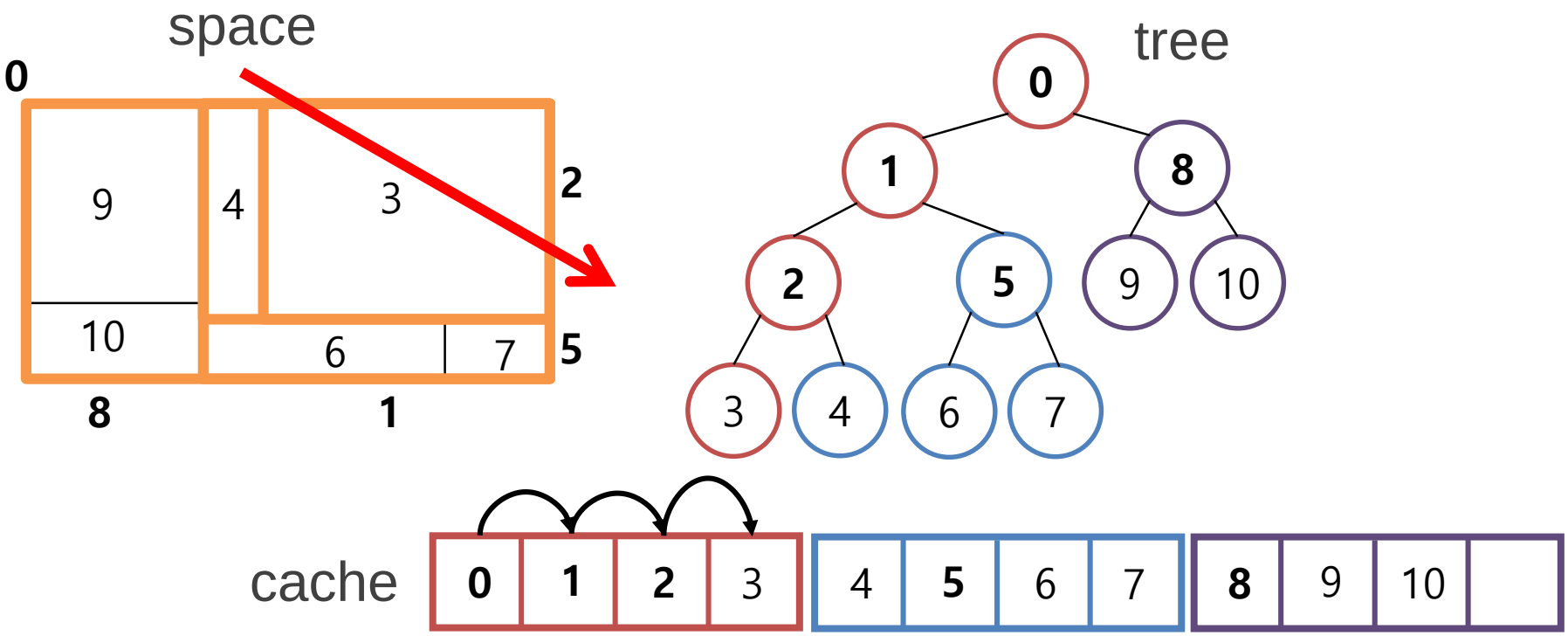
- Our approach
  - The probability of a ray intersecting with a node is proportional to its surface area [Macdonald and Booth 1990]
  - Change the arrangement criterion of child nodes : geometric position → surface area

Traditional Depth-First Layout



- Child nodes are arranged by their geometric positions [Pharr and Humphreys 2010]
  (left node ≤ split plane ≤ right node)

Ordered Depth-First Layout



- 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.*

	SaarCOR [Schmittler et al. 2004]	RPU [Woop et al. 2005]	D-RPU [Woop 2007]	Ours
FP ADD	4	4	16	1
FP MUL	0	4	16	1
FP RCP	4	0	4	0
Stack entry	32	32	32	4
Peak throughput	4	4	4	1
Architecture	SIMD	SIMD	SIMD	MIMD
AS	kd-tree	kd-tree	B-KD tree	kd-tree
Special feature			node update	ODFL

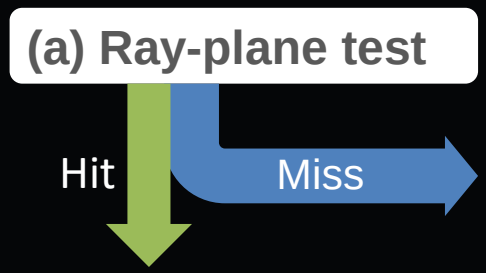
Outline

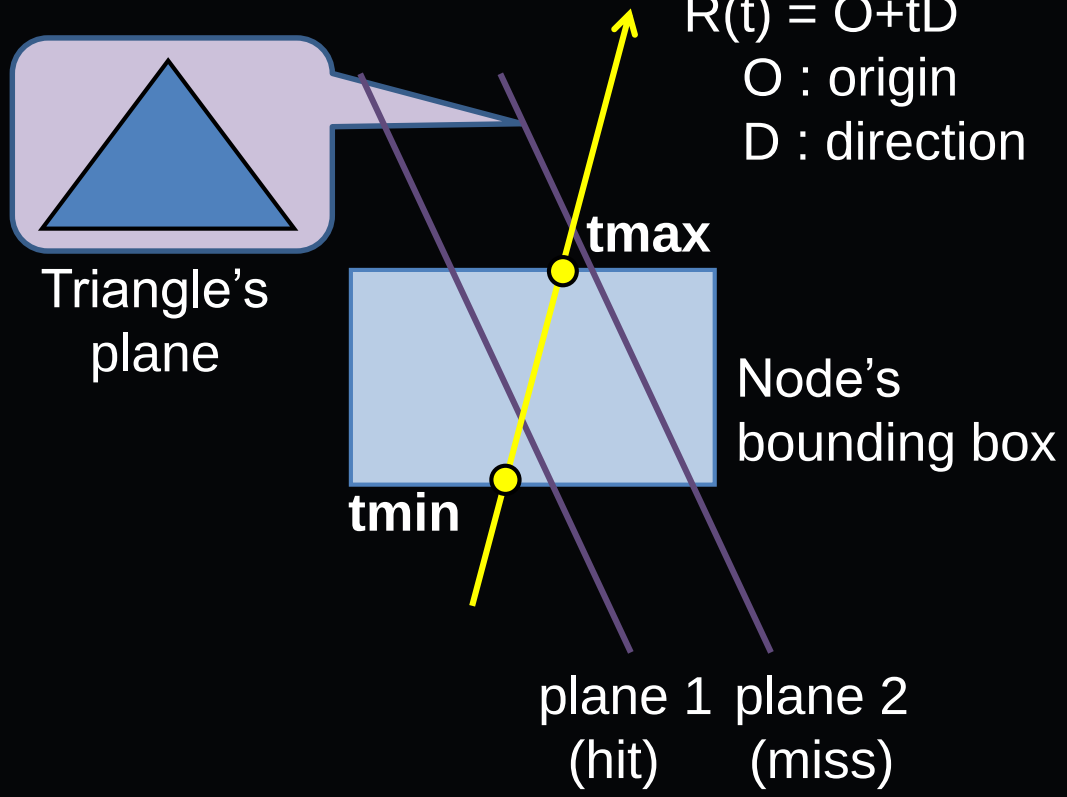
- Introduction and related work
- Overall system architecture
- Traversal with an ordered depth-first layout
- Three-phase intersection test unit
- Ray accumulation unit for latency hiding
- Simulation results and analysis
- Conclusions and future work

Ray-Triangle Intersection Test

- Three-phase calculations

  (a) Ray-plane test

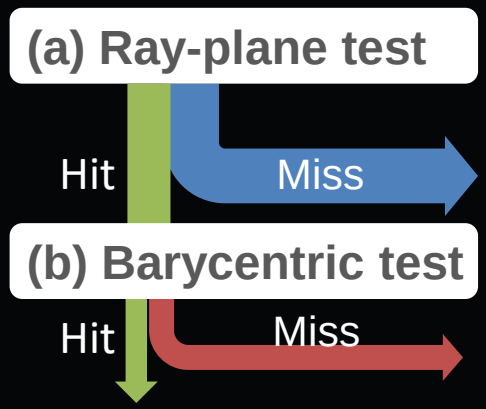


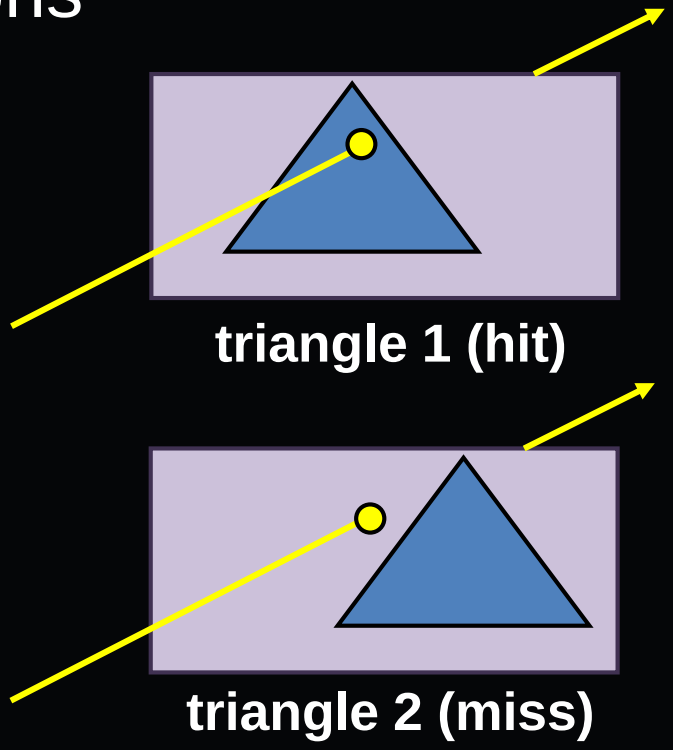


Ray-Triangle Intersection Test

- Three-phase calculations

  (b) Barycentric test

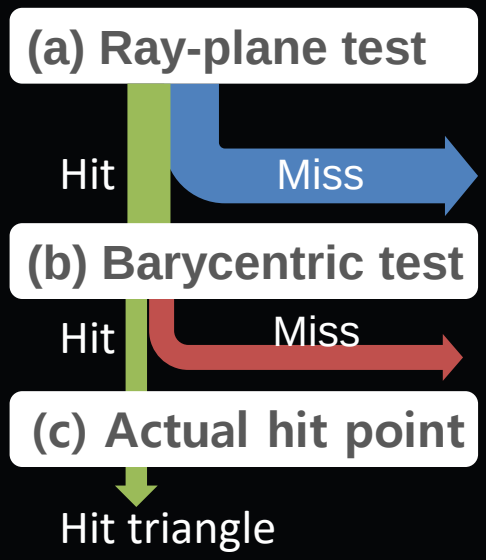


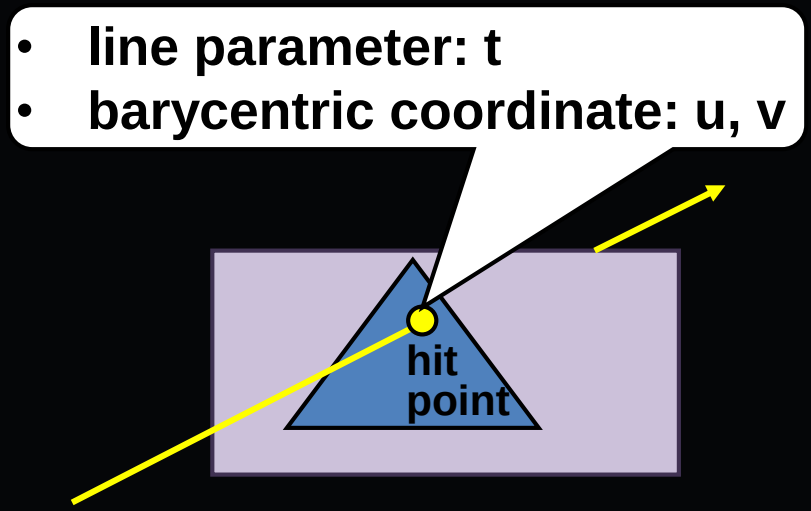


Ray-Triangle Intersection Test

- Three-phase calculations

  (c) Final hit point calculation

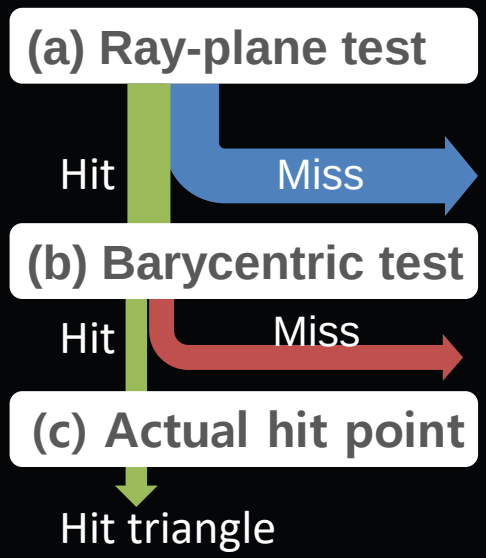


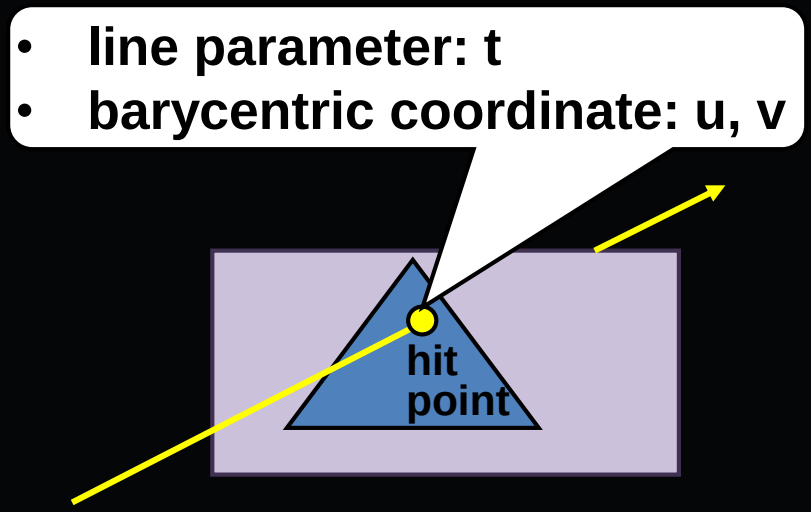


Ray-Triangle Intersection Test

- Three-phase calculations

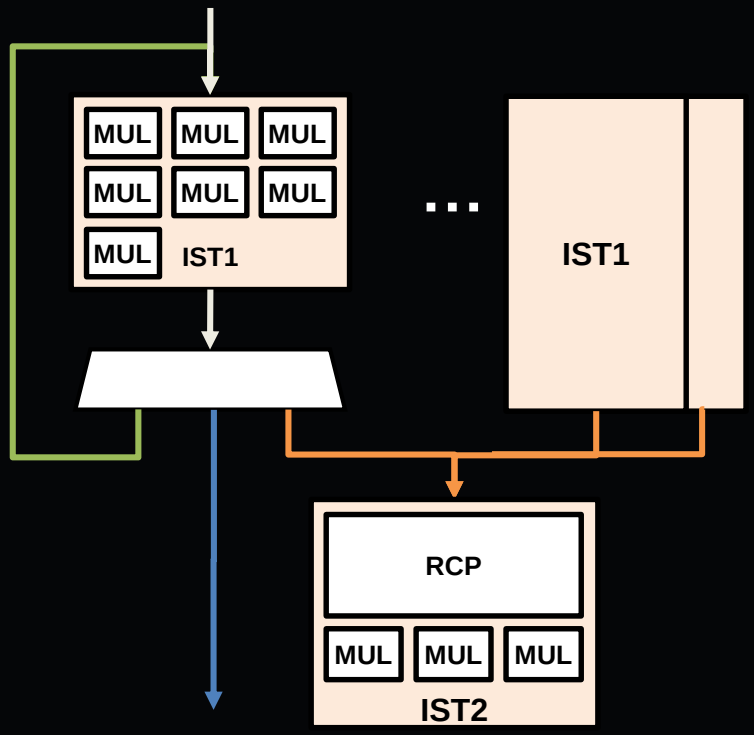
  (c) Final hit point calculation





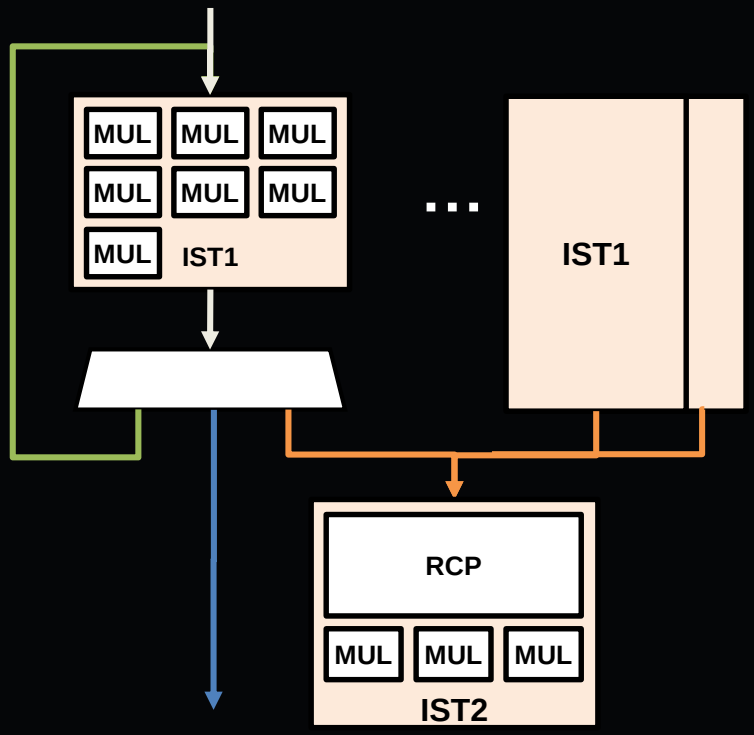
- Previous one-phase architectures [Schmittler et al. 2004; Woop et al. 2005; Kim et al. 2007] do not use this property

Three-Phase Intersection Test Unit



- IST1s perform
  - (a) a ray-plane test
  - (b) a barycentric test
    - If the ray does not pass either process, further computation and memory requests are stopped

Three-Phase Intersection Test Unit



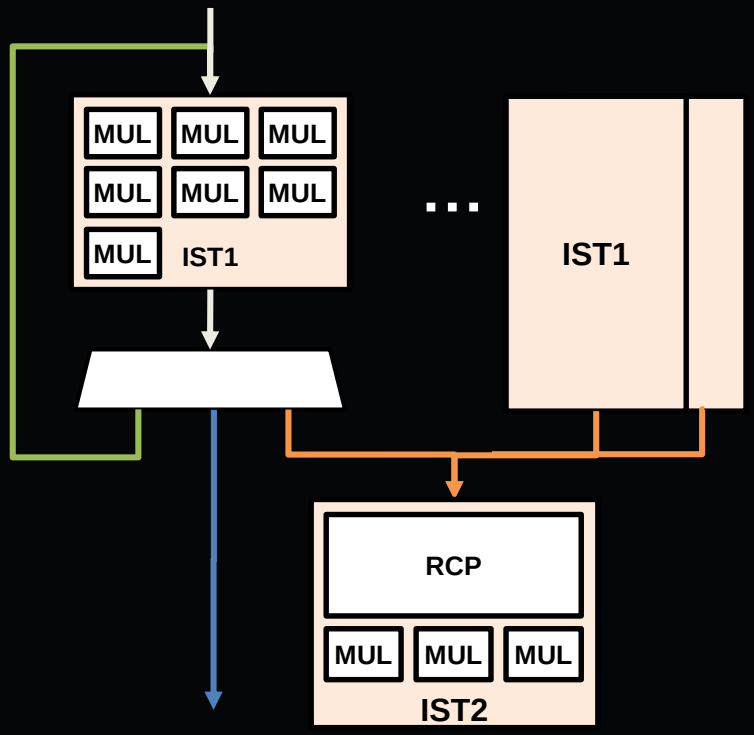
- IST1s perform
  - (a) a ray-plane test
  - (b) a barycentric test
- IST2 performs
  - (c) final hit point calculation

Case 1 (miss)

Case 2 (pass the process (a))

Hit triangle

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

	SaarCOR [Schmittler et al. 2004]	D-RPU [Woop 2007]	CDE [Kim et al. 2007]	Ours IST1	Ours IST2
FP ADD	12	17	12	7	0.375
FP MUL	11	21	27	7	0.375
FP RCP	1	1	1	0	0.125
Throughput	0.8	0.5	1.0	0.76	
Algorithm	[Wald 2004]	[Möller and Trumbore 1997]		[Shevtsov et al. 2007]	

Outline

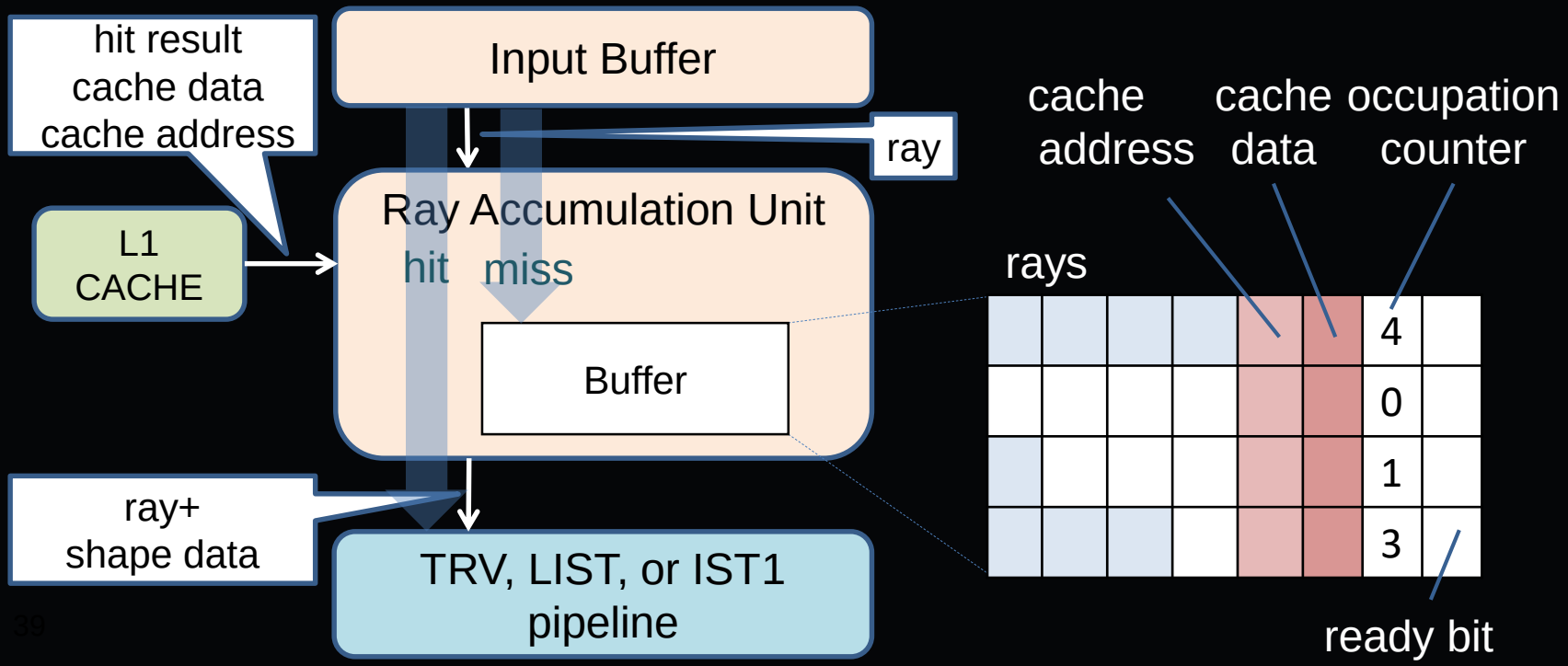
- Introduction and related work
- Overall system architecture
- Traversal with an ordered depth-first layout
- Three-phase intersection test unit
- Ray accumulation unit for latency hiding
- Simulation results and analysis
- Conclusions and future work

Background

- Each of the T&I steps requires memory access to obtain the shape data → 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



Comparison to Existing H/W multi-threading

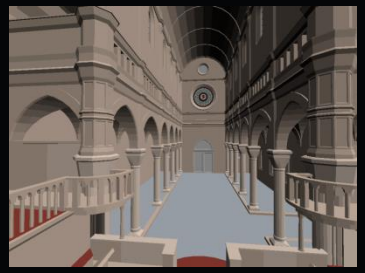
- Small working set size
  - A 32-entry RA buffer requires 4 KB of memory
- Effectively exploit temporal locality
  - The period between the ray's shape data fetching is shorter than existing H/W multi-threading
  - This feature results from our architecture's small working set size

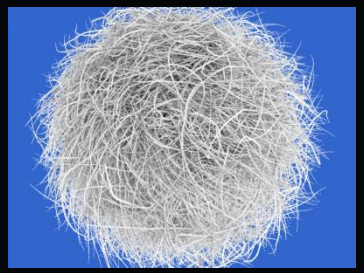
Outline

- 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.*

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

Area Estimation

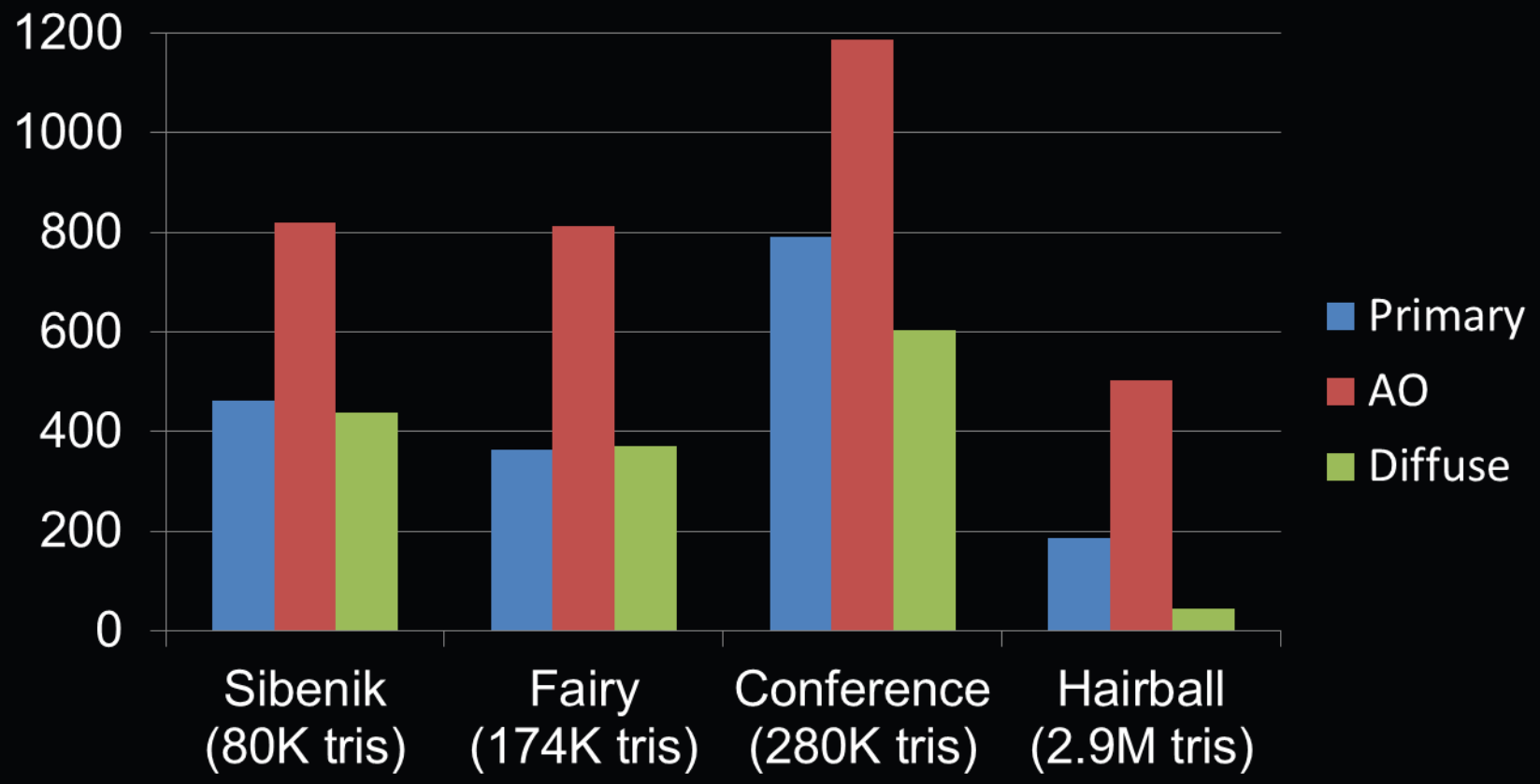
- 500 MHz, 65 nm process, 200mm^2 die size
- Reference: [Woop 2007; Muralimimanohar et al. 2007; Mahesri et al. 2008; Spjut et al. 2009; Kopta et al. 2010]

Table 10: *Area estimates of a T&I core.*

Functional Unit	Area (mm^2)	Total Area (mm^2)	Memory Unit	Area (mm^2)	Total Area (mm^2)
FP ADD	0.003	0.26	TRV L1	0.03	0.72
FP MUL	0.01	0.92	LIST L1	0.028	0.17
FP RCP	0.11	0.22	IST1 L1	0.082	0.66
FP CMP	0.00072	0.08	TRV L2		0.64
INT ADD	0.00066	0.01	LIST L2		0.24
Control/Etc.		0.35	IST1 L2		0.64
			4K RF	0.019	2.26
Wiring overhead					4.95
Total					12.12

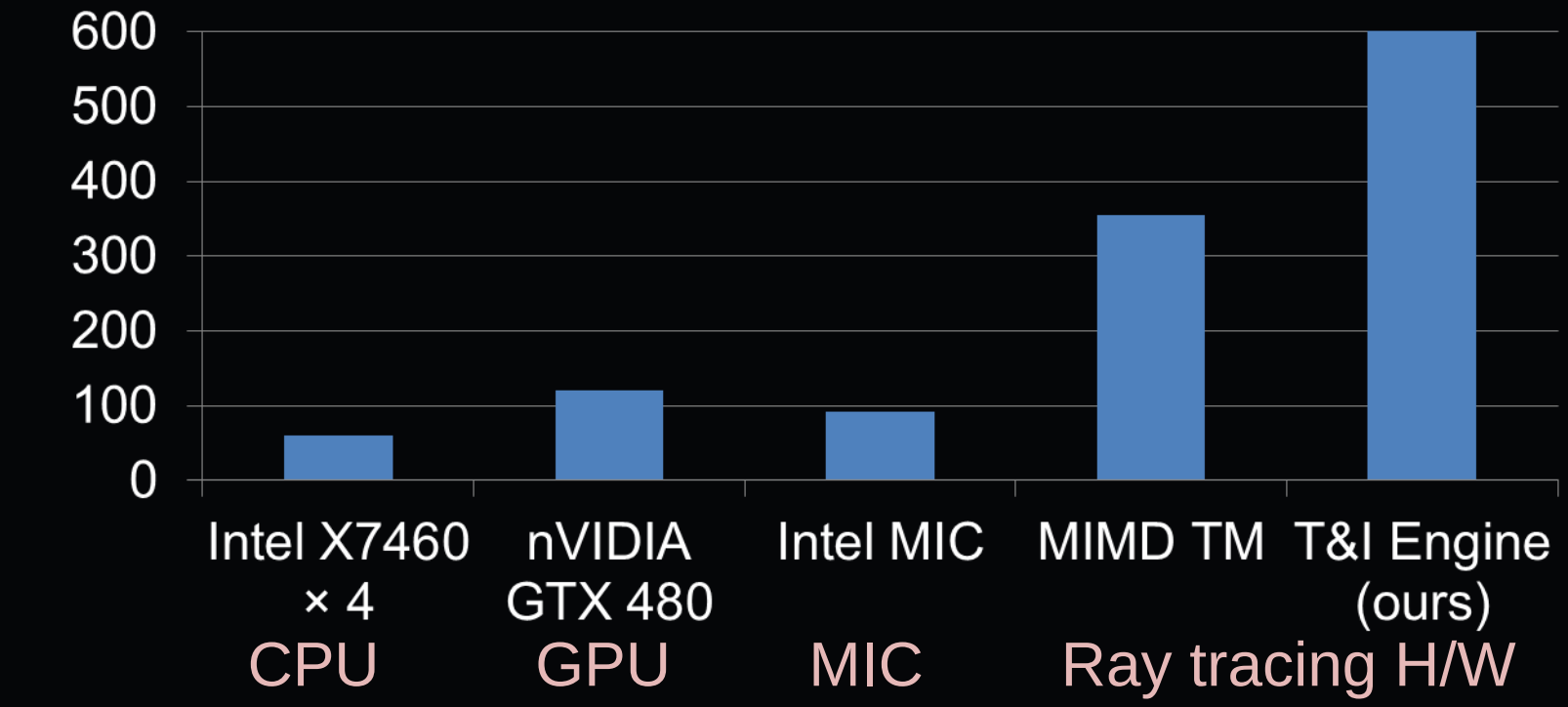
Cycle-Accurate Simulation Results

- Assumption: four T&I cores (48.50 mm^2)



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?