A Novel Mobile GPU Architecture based on Ray Tracing

Won-Jong Lee[1], Youngsam Shin[1], Jaedon Lee[1], Jin-Woo Kim[2], Jae-Ho Nah[2],
Hyun-Sang Park[3], Seokyoon Jung[1], and Shihwa Lee[1]
SAIT Samsung Electronics[1], Yonsei University[2], Kongju National University[3], Korea

***Abstract*-- Recently, with the increasing demand for photorealistic graphics and the rapid advances in desktop CPUs/GPUs, real-time ray tracing has attracted considerable attention. Unfortunately, ray tracing in the current mobile environment is difficult because of inadequate computing power, memory bandwidth, and flexibility in mobile GPUs. In this paper, we present a novel mobile GPU architecture called the SGRT (Samsung reconfigurable GPU based on Ray Tracing) with the following features: 1) a fast compact hardware engine that accelerates a traversal and intersection operation, 2) a flexible reconfigurable processor that supports software ray generation and shading, and 3) a parallelization framework that achieves scalable performance. Experimental results show that the SGRT can be a versatile graphics solution, as it supports compatible performance compared to desktop GPU ray tracers.**

I. INTRODUCTION

Ray tracing is a physically correct rendering algorithm efficiently modeling the interaction between objects and lights, which produces highly realistic graphics images. Due to the requirements of massive computing power and memory bandwidth, ray tracing has been mainly used in off-line rendering field. However, recent rapid advances in desktop CPUs/GPUs and a variety of researches have made real-time ray tracing possible [1]. As a result, the ray tracing is expected to be a new graphics paradigm to create a new market in near future [2].

Mobile graphics has been another trend introducing a new user experiences. Mobile devices are widely used all over the world, and these platforms provide an opportunity creating the new graphics applications. Increased interest in mobile graphics can be seen in the activities of industry standard like OpenGL|ES. In order to maximize user experience, ray tracing is expected to be demonstrated on the mobile devices in near future.

Though mobile graphics capabilities and performance have advanced considerably in recent years, real-time ray tracing in current mobile GPU is very difficult due to the following reasons. First, computational power is inadequate. Ray tracing of a real-world application at HD resolution requires the performance of 300Mray/sec (about 1~2TFLOPS) is needed [3], but the peak performance of current flagship mobile GPU is no more than 256GFLOPS (ARM Mali T658 [4]). Second, mobile GPU lacks efficient branching supports. Ray tracing is a control-flow-intensive algorithm, but mobile GPU cannot fully support branches with limited stack memory. Third, execution model of the mobile GPU is multithreaded SIMD which is not suited for ray tracing, because it causes a divergent branching and memory access in secondary rays. These incoherent rays can lead to a poor SIMD efficiency.

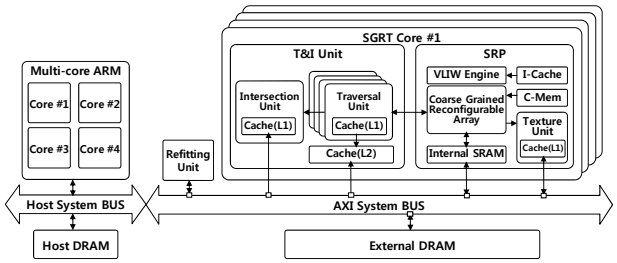

Fig. 1. Our system architecture including the SGRT cores and host processor

In this paper, we propose a new mobile GPU architecture, called SGRT (Samsung reconfigurable GPU based on Ray Tracing), which can solve the problems previously mentioned. The SGRT has three key features. First, it has a fast compact hardware engine that accelerates a traversal and intersection (T&I) which are computationally dominant operations in ray tracing. Second, it employs a flexible reconfigurable processor that supports software ray generation and shading (RGS). Third, it exploits a parallelization framework with real-time operation system (RTOS) that achieves scalable performance.

In addition, our system architecture is designed for recent application processor (AP) that integrates CPUs, GPUs, and DSPs into a single chip with SoC technology. We assign the major modules of the ray tracing into the appropriate computing resources of AP, which is a combination of the tree-rebuild module to reconstruct whole acceleration data structures (on multi-core CPUs), the tree-refit module to update only the changed nodes in the tree (on dedicated H/Ws) and the rendering (on the SGRT). Experimental results show that our GPU can be a versatile graphics solution for future application processor by exposing equivalent performance of recent desktop GPU ray tracers.

II. SGRT CORE ARCHITECTURE

Figure 1 shows the overall system architecture including the SGRT cores and host CPUs. This section describes our architecture in detail.

A. Dedicated Hardware for Traversal and Intersection

The lack of computational power and memory bandwidth of current mobile GPUs motivated us to design a dedicated hardware. Our H/W, called T&I engine, consists of multiple traversal and intersection units with multi-level caches for efficient memory usage, which is similar with previous ray tracing architecture [5][6]. But, unlikely the previous works, our H/W is optimized for mobile environment with the following features. First, it has a smaller area (3.89 mm^2 per core,

Fig. 2. Rendered images by the SGRT simulator: *Ferrari* (left, 210K triangles) and *Fairy* (right, 170K triangles).

65nm). For processing dynamic scenes, our H/W uses bounding volume hierarchies (BVH) that is an object hierarchy, which negates the need for LIST units to manage primitives. In addition, the traversal unit performs both BVH traversal and an intersection test between the ray and the primitive's axis-aligned bounding box (*primAABB*), which can significantly save the area. Second, we minimize the SRAM usage by employing short-stack based traversal algorithm [7]. Third, we combine a primAABB and pre-computed triangle data (*triAccel*) into a 32-byte aligned compact data, which increase the cache efficiency.

High performance features of our previous work [6] like the MIMD architecture for incoherent rays and a ray accumulation unit for latency hiding are directly reused in our H/W. Moreover, we can selectively utilize a specific BVH between the variants (e.g. Full SAH, Binned, SBVH, and LBVH) that are supported by the T&I engine.

B. Reconfigurable Processor for Shading

We utilize a proprietary low-power DSP core developed in our previous work [8][9]; it is called the SRP (Samsung Reconfigurable Processor). The SRP is very flexible for supporting full programmability; thus, various shaders (e.g. material and illumination) can be easily implemented. Unlike the conventional mobile GPU, the VLIW engine of the SRP can fully support control-flow such as recursion and branch, which make recursive ray tracing possible. In addition, the SRP is capable of highly parallel data processing. The coarse-grained reconfigurable array (CGRA) of the SRP makes full use of the software pipeline technique to allow loop acceleration. Therefore, the ray packet stream processing can be done in ray generation and shading kernels, which maximizes the utilization of the functional units.

C. Parallelization Framework

For scalable performance, we built a parallelization framework based on the Samsung Multi-platform Kernel (SMK) [10], a real-time operating system for embedded system. The SMK supports multi-tasking by systematic scheduling in the task queues, and it allows developers to create and use tasks easily. We define an individual task for each SGRT core that is responsible for different pixels (or pixel tiles), then the scheduler can distribute the next tasks to the idle SGRT core first, which results in dynamic load balancing. According to preliminary experiments, we could determine the performance scalability; 3.8x speedup on 4 SGRT cores compared to a single core.

III. Experimental Results

The validity of the SGRT is verified and its performance is evaluated during cycle accurate simulation. The Ferrari and Fairy has been thoroughly tested (Figure 2). Table 1 lists the performance results of ray tracing performed by the SGRT (4 cores), including shadow, reflection and refraction with WVGA (800x480) resolution at 1GHz clock speed. We achieve around 170M RPS (T&I engine), 255M RPS (SRP) and 87.82 fps (Fairy), which may be equivalent to the performance of recent desktop GPU ray tracers (~300M RPS).

TABLE I
PERFORMANCE RESULTS OF THE SGRT ARCHITECTURE

Scene	# of tri.	# of ray	T&I engine				SRP	FPS
			Pipeline utilization	TRV $ hit ratio	IST $ hit ratio	MRPS*	MRPS*	
Fairy	170K	1.7M	87.27	93.83	96.53	171.32	255.72	87.82
Ferrari	210K	1.5M	79.75	92.56	92.92	122.48	319.56	67.83

*MRPS (Mega Rays Per Second)

IV. Conclusion

In this paper, we propose a novel mobile GPU based on ray tracing. This is a first approach to realize a real-time ray tracing in mobile environment, which has been impossible in state-of-the-art OpenGL-based mobile GPU due to the inadequate computational power and memory bandwidth. Furthermore, our system architecture is carefully designed to suit for mobile SoC platform. Simulation results show that our GPU can be a versatile graphics solution by presenting equivalent performance of recent desktop GPU ray tracers. We are now implementing the T&I engine at the RTL level, and we will release the complete GPU product targeted for future consumer electronics such as smart phone, tablet PC, and smart TV.

References

[1] I. Wald, W. Mark, J. Gunther, S. Boulos, and T. Ize, "State of the art in ray tracing animated scenes," *Computer Graphics Forum*, vol. 28, no. 6, pp. 1691-1722, 2009.

[2] H. Jim, "Ray tracing goes main stream," *Intel Technology Journal*, vol. 9, no. 2, pp. 99-108, 2005.

[3] P. Slusallek, "Hardware architectures for ray tracing," *ACM SIGGRAPH*, Course Notes, 2006.

[4] ARM Mali-T658 http://www.arm.com/products/multimedia/mali-graphics-hardware/mali-t658.php, 2012.

[5] S. Woop, J. Schmittler, and P. Slusallek, "RPU: a programmable ray processing unit for real-time ray tracing," *ACM Transactions on Graphics (SIGGRAPH)*, vol. 24, no. 3, pp. 434-444, 2005.

[6] J.-H. Nah, J.-S. Park, C.-M. Park, J.-W. Kim, Y.-H. Jung, W.-C. Park, T.- D. Han, "T&I engine: traversal and intersection engine for hardware accelerated ray tracing," *ACM Transactions on Graphics (SIGGRAPH ASIA)*, vol. 3, no. 6, article 160, pp. 1-10, 2011.

[7] S. Laine, "Restart trail for stackless BVH traversal," *ACM Conference on High Performance Graphics*, pp. 107-111, 2010.

[8] W.-J. Lee, S.-O. Woo, K.-T. Kwon, S.-J. Son, K.-J. Min, C.-H. Lee, K.-J. Jang, C.-M. Park, S.-Y. Jung, and S.-H. Lee, "A scalable GPU architecture based on dynamically embedded reconfigurable processor," *ACM Conference on High Performance Graphics*, poster, 2011.

[9] W.-J. Lee, S.-Y. Jung, and S.-H. Lee, "An effective task scheduling scheme for multicore tile based rendering GPU," *ACM Conference on High Performance Graphics*, poster, 2012.

[10] Y. Shin, S.-W. Lee, M.-Y. Son, and S.-H. Lee, "Predictable multithread scheduling with cycle-accurate thread progress monitor," *ACM Symposium on Applied Computing (SAC '11)*, pp. 627-628, 2011.