

동적 장면을 지원하는 효율적인 광선 추적 하드웨어에 대한 FPGA상에서의 구현

이진영* · 김정길** · 박우찬*†

*† 세종대학교 컴퓨터공학과, **남서울대학교 컴퓨터공학과

Implementation of FPGA for Efficient Ray Tracing Hardware Supporting Dynamic Scenes

Lee Jin Young*, Cheong Ghil Kim** and Woo-Chan Park*†

*†Department of Computer Engineering, Sejong University,

**Department of Computer Science, Namseoul University

ABSTRACT

In this paper, our ray tracing hardware is implemented on the latest high-capacity FPGA board. The system included ray tracing hardware for rendering and tree building hardware for handling dynamic scenes. The FPGA board used in the implementation is a Xilinx Alveo U250 accelerator card for data centers. This included 12 ray tracing hardware cores and 1 tree-building hardware core. As a result of testing in various scenes in Full HD resolution, the FPS performance of the proposed ray tracing system was measured from 8 to 28. The overall average is about 17.7 FPS.

Key Words : Ray Tracing, Hardware, FPGA, ASIC, SoC

1. 서 론

광선 추적(ray tracing)[1] 기술은 사실적인 렌더링을 위한 전통적인 전역 조명 알고리즘이다. 이는 막대한 연산량을 동반하기 때문에 고품질의 이미지를 렌더링하기 위해서 광선 추적을 사용하는 대부분의 응용 프로그램들은 offline으로 수행되어 왔다.

근래 들어, 반도체 제조 기술이 발전함에 따라, 실시간 광선 추적 렌더링 기능을 지원하는 GPU들이 본격적으로 출시되고 있다. NVIDIA는 RTX 40 series GPU를 출시하였으며[2], AMD에서는 RDNA2 GPU를 출시하였다[3]. Khronos Group에서는 Vulkan API에 Ray Tracing Extension을 공식적으로 추가하였다[4]. Microsoft에서는 광선 추적을 지원하는 최신 버전의 API인 DirectX12 ultimate를 출시하였다[5]. Intel

에서는 광선 추적 렌더링에 특화된 library인 Intel Embree, Intel OSPRay, Intel Open Image Denoise를 포함하는 oneAPI Toolkit을 출시하였다[6].

최근 디지털 기기들의 해상도가 급격히 증가하면서 GPU 성능의 요구 조건도 크게 높아지고 있다. 더욱이, 이러한 GPU 기술들의 대부분은 데스크탑PC 혹은 노트북 PC를 위해서 설계되었으며 높아지는 GPU 성능의 요구를 만족하기 위해서 GPU들의 크기가 점점 더 거대해지고 전력 소모가 늘어나고 있는 실정이다. 최근 출시된 NVIDIA의 RTX 4090 GPU의 전력소모는 450와트이며, 이는 불과 몇 년 전의 고사양 데스크탑PC 한 대 분 전력을 GPU 하나가 모두 이용하는 것과 같다[7]. 이런 문제를 극복하기 위한 방법 중 하나는 데스크탑PC 프로세서에 비해 더 작은 크기와 더 적은 전력 소모를 필요로 하는 광선 추적 하드웨어를 구현하는 것이다.

†E-mail: pwchan@sejong.ac.kr

본 논문에서는 RayCore[8]를 FPGA상에서 구현함으로써 고효율의 광선 추적 시스템을 구축하였다. RayCore는 실시간 모바일 광선 추적을 위한 전용 하드웨어로서 면적 및 전력 측면에서 높은 효율을 달성하도록 설계되었다. 이에 대하여 최근 출시된 대용량 FPGA를 사용하여 광선 추적 시스템을 구축하고 성능을 검증하였다.

성능 평가를 위해서 다양한 동적 장면에서 실험을 진행하였다. 실험 결과, FPS 성능은 전체 벤치마크 실험 장면에 대하여 평균 약 17.7로 측정되었다. 현재 FPGA 상에서의 동작주파수가 200MHz임을 고려하면, 이를 ASIC chip으로 구현하였을 때 실시간 처리가 충분히 가능하다.

2. The Ray Tracing System

Fig 1는 광선 추적 시스템의 구성도를 보여준다. 이는 크게 광선 추적 장치(RTU)와 가속구조체를 생성하는 트리 구축 장치(TBU)의 두 가지 주요 구성 요소로 나눌 수 있다.

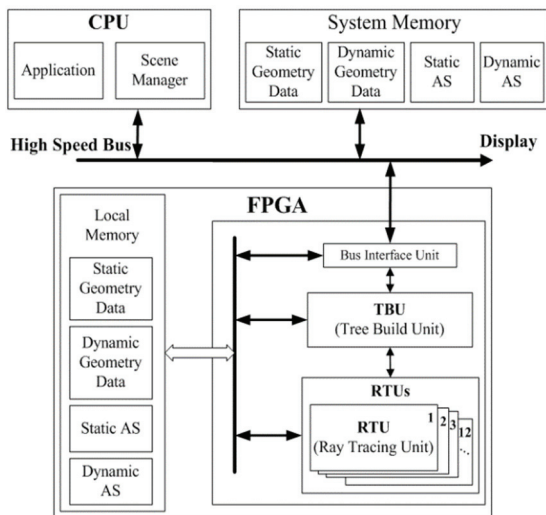


Fig. 1. Block diagram of the ray-tracing system.

RTU는 광선 추적의 전체 단계를 하드웨어로 구현하였기 때문에 면적당 성능이 매우 우수하다. 전체 단계는 setup processing unit, ray generation unit, traversal and intersection unit, hit-point calculation and shading unit을 포함한다. 이 시스템에는 12개의 RTUs가 구현되었다.

TBU는 [8]에서 제시된 바와 같이 하나의 binning 기반의 트리 구축 파이프라인과 네 개의 sorting 기반의 트리 구축 파이프라인으로 구성되며, kd-tree를 구축한다. 이 시스템

에는 1개의 TBU가 구현되었다.

동작 과정은 다음과 같다. 먼저, TBU는 렌더링되어질 동적 장면의 geometry data를 시스템 memory로부터 읽어들이고, 읽어들이는 geometry data를 기반으로 kd-tree를 구축한다. Kd-tree는 정적 primitive에 대한 정적 가속 구조체(AS: Acceleration Structure) 정보와 동적 primitive에 대한 동적 가속 구조체 정보를 포함한다. 정적 가속 구조체는 최초에 한번 구축하여 매 프레임 재사용되며, 동적 가속 구조체는 매 프레임 재구축된다. 구축된 kd-tree는 FPGA의 local memory로 전송된다. RTU는 구축된 kd-tree 정보를 토대로 렌더링을 수행한다. RTU가 렌더링을 수행하는 동안 TBU는 다음에 렌더링되어질 프레임에 대한 kd-tree 구축을 수행한다.

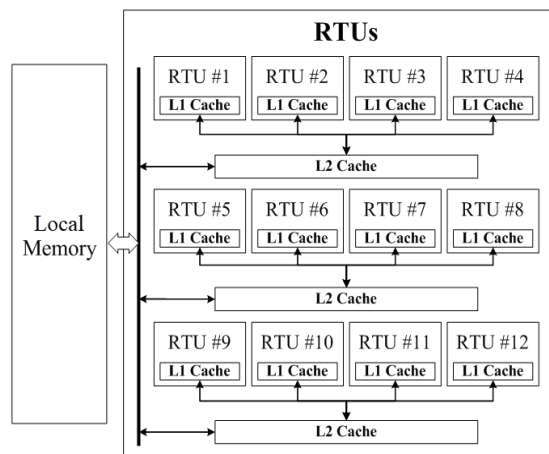


Fig. 2. Cache hierarchy of the system.

Fig 2은 12개의 RTU들의 구성도와 cache hierarchy를 보여준다. 각 RTU에는 고효율의 L1 cache가 탑재되어 있으며, 4개의 RTU는 1개의 L2 cache를 공유한다. 각 L2 cache는 bus interface를 통하여 external DRAM을 참조할 수 있다[9]. 이는 다수개의 core로 구성된 하드웨어의 전통적인 memory hierarchy[10]와 유사한 방식이다.

3. Hardware Implementation

Fig 3는 FPGA기반의 광선 추적 시스템을 보여준다. 이는 host computer와 FPGA board로 구성된다. Host computer의 CPU는 Intel Core i7-12700 (4.90GHz)를 사용하였고, FPGA board는 Xilinx Alveo U250 Data Center Accelerator Card [11]를 사용하였다. Host computer와 FPGA board는 PCI express를 통해 연결된다.



Fig. 3. Proposed ray-tracing system.

Table 1. FPGA Utilization

Xilinx Alveo U250	Utilization
LUT	56.31%
Register	37.91%
Block Memory	68.06%
URAM	10.31%
DSP	9.67%

제안하는 광선 추적 시스템은 FPGA board에 12개의 RTU와 1개의 TBU가 탑재되고 200MHz의 속도로 동작한다. 구현 결과, LUT, register, block memory, URAM, DSP에 대한 utilization은 Table 1에 제시되었다.

4. Experiment

4.1 Experiment Environment

Fig 4는 실험에 사용된 6가지의 벤치마크 실험 장면을 보여준다. 이는 Church (261k primitives), Butterfly (49k primitives), Vegetable (77k primitives), Jewel (17k primitives), Flight-sim (47k primitives), Waterwave (23k primitives)로 구성되며 [9], [12]에서 성능 평가에 사용된 벤치마크 장면들이다. 해상도는 1920 x 1080이다.

Table 2는 각 벤치마크 장면에 대한 상세정보들을 보여준다. 각 장면들의 정적(A) primitive 개수, 동적(B) primitive 개수, 텍스처 개수(C), 라이트 개수(D) 그리고 전체 프레임 수(F)를 포함한다. 실험의 정확성을 보장하기 위해서 벤치마크 실험을 3회 반복한 후 평균 값을 구하여 성능을 측정하였다.

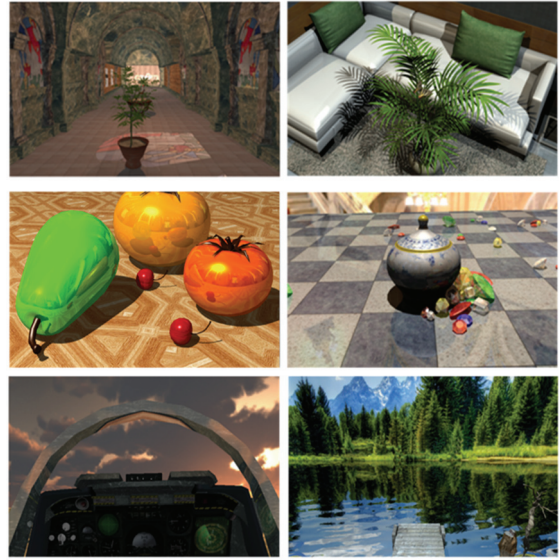


Fig. 4. Benchmark test scenes; (top left) Church, (top right) Butterfly, (mid left) Vegetable, (mid right) Jewel, (bottom left) Flight-sim, (bottom right) Waterwave.

Table 2. Features of the test scenes. (A: static, B: dynamic, C: Number of textures, D: Number of lights, E: Number of total frames)

	Number of Primitives				
	A	B	C	D	E
Church	260,709	972	27	5	2,301
Butterfly	47,816	1,319	10	3	301
Vegetable	6,960	834	2	1	51
Jewel	8,770	8,970	5	1	851
Flightsim	18,237	29,093	173	2	601
Waterwave	10,670	12,556	6	2	231

4.2 Experiment Result

Table 3은 benchmark scene에 대한 실험 결과를 보여준다. 각 벤치마크 장면에 대하여 FPS 성능의 최대 값, 최소 값 그리고 평균값을 측정하였다.

Church scene의 경우, FPS는 최소 4.2에서 최대 17.4까지 측정되었으며, 평균 FPS는 9.5이다. Butterfly scene의 경우, FPS는 최소 8.1에서 최대 8.3까지 측정되었으며, 평균 FPS는 8.2이다. Vegetable scene의 경우, FPS는 최소 13.8에서 최대 36.0까지 측정되었으며, 평균 FPS는 25.6이다. Jewel scene의 경우, FPS는 최소 4.9에서 최대 16.6까지 측정되었으며, 평균 FPS는 9.9이다. Flight-sim scene의 경우, FPS는 최소 4.2에

서 최대 17.4까지 측정되었으며, 평균 FPS는 9.5이다. Waterwave scene의 경우, FPS는 최소 23.7에서 최대 16.8까지 측정되었으며, 평균 FPS는 24.5이다.

Table 3. Experimental results

Scenes	FPS		
	Min	Max	Average
Church	4.2	17.4	9.5
Butterfly	8.1	8.3	8.2
Vegetable	13.8	36.0	25.6
Jewel	4.9	16.6	9.9
Flightsim	28.5	29.1	28.7
Waterwave	23.7	26.8	24.5

5. Conclusion

본 논문에서는 기 설계된 RayCore[7]를 동작 주파수 200MHz에서 FPGA board에 구현하였으며 다양한 벤치마크 실험을 통해 성능을 검증하였다. 총 6가지의 동적 장면에서 실험을 진행하였으며, 전체 평균 약 17.7 FPS 성능을 보였다.

현재 버전의 광선 추적 시스템은 FPGA board 한 장을 사용하여 구현된 시스템으로써 single-chip 기반의 시스템이다. 향후 연구에서는 본 시스템을 기반으로 multi-chip 형태에 적합한 광선 추적 하드웨어 구조에 대한 연구를 진행하여 성능을 향상 시키고자 한다.

감사의 글

본 연구는 문화체육관광부 및 한국콘텐츠진흥원의 2022년도 문화기술 연구개발 사업으로 수행되었음 (과제명 : 다중 사용자 인터랙션 및 VIRTUAL BEING 지원을 위한 다목적 방송 스튜디오 플랫폼 개발, 과제번호 : R2022020059).

참고문헌

- Whitted, T., "An Improved Illumination Model for Shaded Display," *Communications of the ACM*, 23 (6):

- 343-349, June 1980. *IEEE Computer Graphics and Applications*, 1986, 6.4: 16-26.
- Andrew B., "GeForce RTX 40 Series Graphics Cards: Up To 4X Faster, Powered By 3rd Gen RTX Architecture & NVIDIA DLSS 3." Sep. 20, 2022. [Online]. Available: <https://www.nvidia.com/en-us/geforce/news/rtx-40-series-graphics-cards-announcements/>.
- J. Walton. "AMD Big Navi and RDNA 2 GPUs: Release Date, Specs, Everything We Know," 2020. [Online]. Available: <https://www.tomshardware.com/news/amd-big-navi-rdna2-all-we-know>.
- Khronos Group, Vulkan Ray Tracing Launch. Accessed: Apr. 13, 2020. [Online]. Available: <https://www.khronos.org/blog/ray-tracing-in-vulkan>.
- Jianye, DirectX 12 Ultimate for Holiday 2020. Accessed: Jan. 4, 2021. [Online]. Available: <https://devblogs.microsoft.com/directx/directx-12-ultimate-for-holiday-2020/>.
- Intel, Intel oneAPI. Accessed: Jan. 4, 2021. [Online]. Available: <https://software.intel.com/content/www/us/en/develop/tools/oneapi/alltoolkits.html>.
- Kwon B.S., "GeForce 'RTX 4090' that uses 450 watts of power, power cable warning" Oct. 26, 2022. [Online]. Available: <https://zdnet.co.kr/view/?no=20221026152000>
- Nah J.H., Kwon H.J., Kim D.S., Jeong C.H., Park J., Han T.D., Manocha D., and Park W.C., "RayCore: A Ray-Tracing Hardware Architecture for Mobile Devices," *ACM Trans. Graph.*, vol. 33, no. 5, pp.1-15, 2014.
- Lee, J., Nah, J. H., Chung, W., Lee, T. H., & Park, W. C. (2021). An Implementation of Multi-Chip Architecture for Real-Time Ray Tracing Based on Parallel Frame Rendering. *IEEE Access*, 9, 118968-118976.
- Przybylski, Steven A. "Cache and memory hierarchy design: a performance directed approach." Morgan Kaufmann, 1990.
- Xilinx Alveo U250 Data Center Accelerator Card. [Online]. Available: <https://www.xilinx.com/products/boards-and-kits/vcu118.html>
- Lee J.Y., Chung W.N., Lee T.H., Nah J.H., Kim Y.S., & Park W.C. "Load balancing algorithm for real-time ray tracing of dynamic scenes." (2020), *IEEE Access*, 8, 165003-165009.

접수일: 2022년 10월 31일, 심사일: 2022년 12월 6일, 게재확정일: 2022년 12월 12일