ISSN:1226-7244 (Print)
ISSN:2288-243X (Online)
논문번호 17-03-18
j.inst.Korean.electr.electron.eng.Vol.21,No.3,276~279,September 2017
http://dx.doi.org/10.7471/ikeee.2017.21.3.276
103

SIMT 구조 기반 GPGPU를 이용한 고속 Rasterizer 구현

Implememtation of Fast Rasterizer processing using GPGPU based on SIMT structure

김 치 용*

Chiyong Kim*

Abstract

In this paper, SIMT structure based GPGPU (General Purpose Computing on Graphics Processing Units) is used for accelerating the Rasterizer which constitutes the screen of the display device in pixel unit. The GPU has a large number of ALUs, and the processing is very fast because of parallel processing. Therefore, in this paper, we implemented a rasterizer that generates a 3D graphics model using a CPU that performs operations sequentially and a GPU that performs operations in parallel. We confirmed that proposed rasterizer in this paper is 1.45 times better than rasterizer using Intel CPU when generating one frame.

요 약

본 논문에서는 디스플레이 장치의 화면을 픽셀 단위로 구성하는 Rasterizer의 가속화를 위하여 SIMT구조의 GPGPU(General Purpose computing on Graphics Processing Units)를 사용하였다. GPU는 많은 수의 ALU를 가지고 있고, 병렬처리하기 때문에 연산처리가 매우 빠르다. 따라서 본 논문에서는 연산을 순차적으로 수행하는 CPU와 연산을 병렬적으로 수행하는 GPU를 이용하여 3D그래픽스 모델을 생성하는 rasterizer를 구현했다. 한 프레임 생성 시 Intel CPU를 이용한 rasterizer보다 본 논문에서 제안하는 rasterizer가 1.45배 좋은 성능을 확인하였다.

Key words : 3D Graphic processing, Rasterization, GPGPU, thread, SIMT

* Dept. of Computer Science, Seokyeong University
e-mail:kcy@skuniv.ac.kr, tel:02-940-7759

※ Acknowledgment
This research was supported by Seokyeong University in 2017
Manuscript received Sep.13 2017; revised Sep.18 2017; accepted Sep.20, 2017

Ⅰ. 서론

GPU와 display의 발전에 따라 graphic의 응용분야가 급속히 확대되고 있다. 이는 실시간으로 많은 연산량을 가지는 3D 그래픽이 요구되며 있으며, 특히 고급 그래픽 응용에서는 더욱 빨리 처리할 수 있는 여러 알고리즘의 연구가 활발하게 진행되고 있다.

3D 그래픽스는 3개의 정점좌표를 이용하여 하나의 polygon을 구성한다. 그림 1의 (a)를 보면 3개의 점이 있다. 이 세 점이 그림 1의 (b)처럼 polygon을 이룬다. 이러한 polygon을 토대로 그림 1의 (c)처럼 색상을 입히고 색상을 입힌 polygon들

이 모여 입체감 있는 현실적인 3D 이미지 모델을 생성한다. 따라서 많은 수의 polygon이 필요하며 이에 따라 한 frame의 3D 이미지모델을 생성하는 과정에서도 많은 연산이 필요하다. 3D 그래픽스 연산 구조에서 rasterizer는 입력된 3D 모델의 정보를 기반으로 하여 모델의 내부 화소 전체의 색상을 연산하는 연산기로 3D 그래픽스 처리과정에서 가장 많은 연산 시간이 소요되는 단계이다.

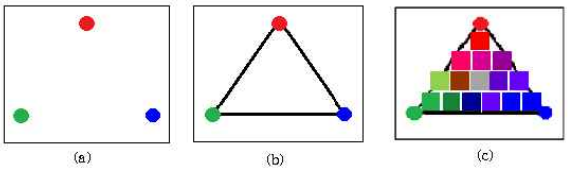

Fig. 1. Creating a polygon and define colors
그림 1. polygon 생성 및 색 상정하기

본 논문에서는 연산량 많은 rasterizer를 256개의 thread를 가지는 GPGPU를 활용해 고속 rasterizer를 구현했다

Ⅱ. Multi thread GPU를 이용한 rasterizer

1. GPGPU의 구조

해당 논문의 GPGPU는 그림 2와 같은 구조를 가진다[1].

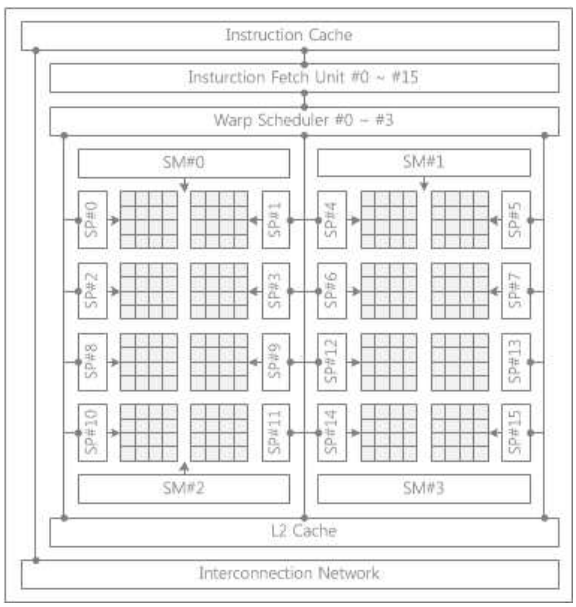

Fig. 2. Architecture of GPGPU
그림 2. Rasterizer 처리 과정

16개의 thread를 warp이라 표현하며 하나의 warp는 하나의 stream processor를 통해 제어된다. 이러한 stream processor 4개를 묶어 stream multiple processor라고 하며 4개의 wrap, 즉 64개의 thread를 제어한다. 이러한 stream multiple processor 4개가 모여 본 논문의 GPGPU를 구성한다[2].

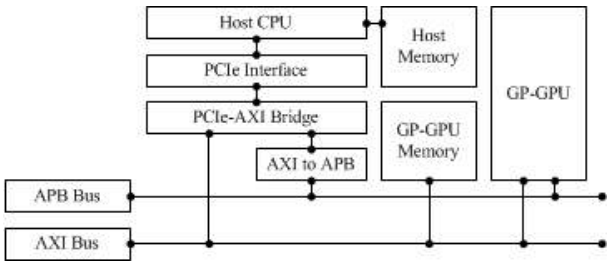

Fig. 3. Interface of GPU
그림 3. GPU의 인터페이스

해당 GPGPU는 그림 3과 같은 인터페이스를 가진다. GPGPU는 AXI버스와 APB버스를 사용하는데 AXI버스는 GPGPU가 실행할 명령어 코드와 GPGPU가 사용할 데이터들의 이동 통로로 사용된다. APB버스는 레지스터 초기화 등의 GPU 제어에 사용된다. AXI 버스는 128 bit의 bit width를 가지며, APB 버스는 32 bit의 bit width를 가진다. Host PC는 PCIe를 통해 GPGPU를 제어한다. PCIe-AXI Bridge를 통해 GPGPU의 메모리에 접근하는지 GPGPU에 접근하는지를 결정하며 AXI 버스와 APB버스의 구분은 접근하는 어드레스에 의해 결정된다. Host PC 관점에서 본다면 Host PC는 Address와 데이터 값을 PCIe 드라이버를 통해 보내주게 된다.

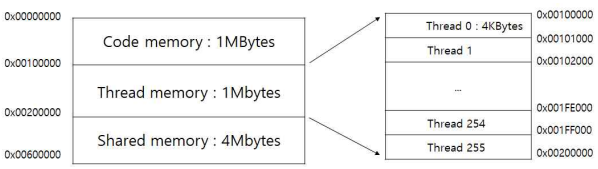

Fig. 4. Memory structure of GPGPU
그림 4. GPGPU 메모리 구조

그림 4은 GPGPU의 메모리 구조이다. GPGPU의 메모리는 그림 4처럼 크게 세 부분으로 나눌 수 있다. code memory에는 해당 프로세서에서 실행되는 바이너리코드가 저장되고, thread memory에는 thread에서 선언한 지역변수, 전역변수들이 저장되며

thread memory의 크기는 thread의 개수와 할당된 메모리 크기에 종속되게 된다. 그림 4를 보면 thread 하나는 4 Kbyte를 차지하며 따라서 thread memory는 1Mbyte가 된다. shared memory는 모든 thread가 접근할 수 있는 메모리 영역이며 데이터의 입출력 통로가 된다. 여러 개의 thread가 동일한 shared memory에 접근하므로 데이터를 동시에 접근하여 write하거나 read하는 경우 문제가 발생할 수 있다. 이러한 문제를 막기 위해 스트림 프로세서에는 Thread ID를 가지는 Special Register가 존재한다.

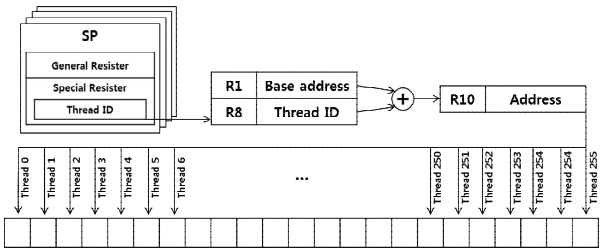

Fig. 5. Memory operation of GPGPU
그림 5. GPGPU의 memory 연산

그림 5는 GPGPU의 memory 연산을 도식화 한 그림이다. SP는 base address를 가지고 있는 register와 thread ID를 가지고 있는 special register의 값을 이용하여 shared memory에 접근할 address를 생성해 shared memory에 접근한다.

이와 같은 GPU를 활용, rasterizer를 구현한다면 고속으로 렌더링 과정을 수행할 수 있게 된다[3].

2. Rasterizer의 처리과정

Rasterizer는 화면을 구성하는 pixel 단위의 색상을 처리하는 프로세서로 3D 그래픽 파이프라인에서 중요한 역할을 한다. Rasterization의 수행과정은 그림 6와 같이 geometry 단계의 결과로 얻은 정점 정보를 polygon 형태로 변환하는 primitive assembly 단계, 화면에 칠해질 화소의 색상을 결정하는 generate pixel mask 단계, 칠해질 화소의 색상을 연산을 통하여 구하는 interpolation까지 총 3단계로 구성된다.[4]

Fig. 6. Processing of Rasterizer
그림 6. Rasterizer 처리 과정

Primitive Assembly와 Generate Pixel Mask 과정은 Host PC에서 처리하고 Interpolation 과정 중 RGB 버퍼를 클리어 하는 과정을 GPGPU를 통해 처리한다. RGB 버퍼를 클리어 하는 과정은 연산은 단순하지만 VGA 해상도라면, 307,200 개의 픽셀을 가지며 픽셀 당 3개의 채널을 가지므로 921,600 개의 연산이 필요하게 된다. 해당 연산을 GPGPU를 사용해 병렬 처리한다면 빠르게 처리할 수 있게 된다. 921,600의 연산을 256개의 스레드를 가지는 GPGPU를 사용해서 처리하게 된다면 스레드 하나 당 3,600 번의 연산을 수행하게 된다. GPGPU의 처리 과정을 도식화 하면 그림 7과 같다.

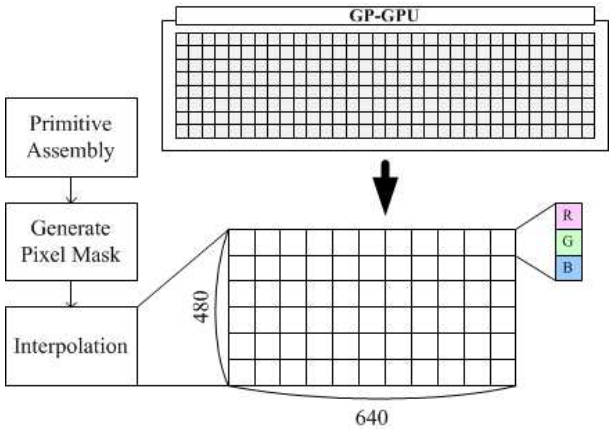

Fig. 7. processing of GPGPU
그림 7. GPGPU의 처리 과정

III. 실험 결과 및 결론

1. 실험 결과

실험에는 GPGPU와 Intel CPU를 가지고 VGA 해상도를 가지는 그래픽 모델에 대해 rasterization 연산을 비교했다.

실험에서 사용한 Intel CPU와 그에 대한 성능은 다음 표 1과 같다.

Table 1. Specifications of Intel CPU
표 1. Intel CPU의 성능

CPU	RAM
Intel i7-4470 / 3.4 GHz	DDR3 8GB

실험에서 사용한 GPGPU는 Xilinx VC-707 FPGA를 사용하여 합성하였고 동작 주파수와 합성 결과는 다음 표 2와 같다[5].

Table 2. Stnthesis result of GPGPU
표 2. GPGPU의 합성 결과

Resource	Utilization
GPGPU clock	100 Mhz
LUT	178,719
LUTRAM	11,099
FF	128,863
BRAM	106
DSP	112

그림 8은 GPGPU를 통해 생성한 그래픽 모델에 대한 결과 이미지 이다.

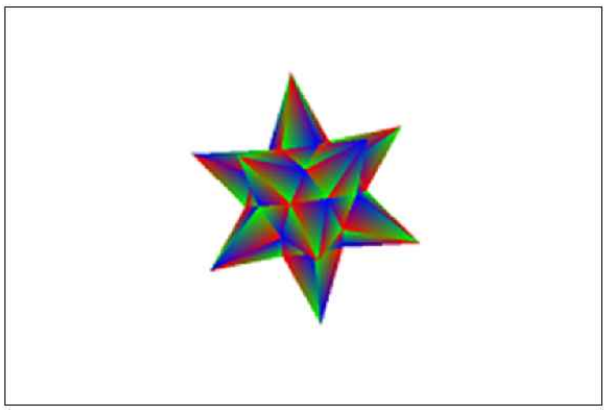

Fig. 8. 3D model created with Rasterizer
그림 8. Rasterizer로 생성된 3D 모델

표 3은 Intel CPU와 GPGPU의 한 프레임 처리 결과를 나타내는 표이다. Intel CPU는 3.4 GHz의 동작 주파수를 가지나 GPGPU는 100Mhz의 동작 주파수를 가진다. 그에 비해 성능은 약 1.45배 빠르다. Intel CPU는 한 픽셀 당 약 5.20 ns의 시간이 걸리며 GPGPU는 3.58 ns의 시간이 걸린다.

Table 3. Experimental result
표 3. 실험결과

Rasterizer	Processing Time	Processing Time per pixel
Intel CPU	1.6 ms	5.20 ns
GPGPU	1.1 ms	3.58 ns

2. 결론

본 논문에서는 256개의 스레드를 가지는 GPGPU를 활용하여 3D 그래픽 처리 알고리즘인 rasterization을 구현했다. rasterization 에서 interpolation단계에서 RGB 버퍼를 클리어 하는 단계를 GPGPU를 통해 멀티 스레드로 동작 시켰다. 해당 rasterizer의 성능을 확인하기 위해 VGA 해상도의 3D 모델을 가지고 실험했다. VGA 해상도의 이미지는 307,200 개의 pixel을 가지며, RGB 채널을 가지므로 921,600 개의 연산을 필요로 한다. 256개의 thread를 가지는 GPU를 통해 연산을 한다면 하나의 스레드는 3,600 개의 연산을 수행해야 한다. 3.4GHz의 동작 주파수를 가지는 Intel CPU와 비교를 했으며 약 1.45 배의 성능 향상을 확인 했다.

References

[1] Kwanho Lee, “A Design of a SIMT architecture base GP-GPU based for parallel acceleration of algorithms”, Master thesis, Seokyeong University, 2017

[2] Kwanho Lee, Chi-yong Kim, “A Design of a High Performance Stream Processor without Superscalar Architecture”, *j.inst.Korean.electr.electron.eng,* Vol 21. No 1.

[3] Kwanho Lee, Chi-yong Kim, “Thread Distribution Method of GP-GPU for Accelerating Parallel Algorithms”, *j.inst.Korean.electr.electron.eng,* Vol 21. No 2.

[4] Dohyun Kim, “A Design of a Multi-threaded Graphic Pipeline and Rasterizer for accelerating GP-GPU”, Master thesis, Seokyeong University, 2016

[5] http://www.xilinx.com