논문 2004-41SD-11-14

휴대형기기에 적합한 내장형 3차원 그래픽 렌더링 처리기 설계

(A design of The Embedded 3D Graphics Rendering Processor for Portable Devices)

우 현 재*, 장 태 홍**, 이 문 기***

(Hyun Jae Woo, Tae Hong Jang, and Moon Key Lee)

요 약

기존의 3차원 그래픽 가속기는 전력소모 및 규모가 커서 휴대형 기기에는 적합하지 않다. 따라서 본 논문에서는 휴대형기기에 적합한 저전력 소규모의 3차원 렌더링 처리기를 제안한다. 소규모의 구현을 위하여 반복연산 및 고정소수점 연산을 사용하였다. 또 저전력의 고려를 위해 텍스쳐 유무에 따라 효율적으로 파이프라인을 관리하였고, 삼각형 셋업 및 에지워킹 단은 순차적으로, 3차원 영상 가속기의 성능을 좌우하는 스캔라인처리와 스팬처리 단은 병렬적으로 처리하게 설계하였다. 설계한 렌더링 처리기는 800*600의 해상도 지원과 32비트의 트루칼러를 지원하며 0.25μm ASIC공정에서는50MHz로 동작하여 초당 47.88M 개의 픽셀과 33.25 프레임을 처리하며 텍스쳐 매핑을 포함 64만 게이트를 가지며 면적은4.9827mm*4.9847mm 이이며 파워소모는 263.7mW이다.

Abstract

This paper proposes 3D graphics accelerator, especially rendering unit, for portable devices. The existing 3D architecture is not suitable for portable devices because of its huge size. To reduce the size, we use iterative architecture and fixed-point calculation. In this paper, we suggest the format of fixed-point comparing with the result images, and some special technique to control. Finally, it is implemented with FPGA and 0.25um ASIC technology respectively. The ASIC chip can execute 47.88M pixels per second. The size of ASIC chip is 4.9287mm*4.9847mm and the power consumption is 263.7mW with 50MHz operation frequency.

Keywords: 3D Graphics, Rendering, Geometry, Rasterizer, Texture mapping

I. 서 론

최근 디지털 컨버전스 (digital media convergence)의 개념이 확산됨으로 인하여 디지털 미디어에 여러 가지 기능들이 통합되고 있는 추세이다. 이러한 추세는 개인용 Portable Device에서도 예외가 아니다. 이전의 Portable Device들에서는 단순한 텍스트 위주의 문서나 정보를 위한 응용 프로그램 수준의 서비스를 제공하였으나, 최근에는 발달된 Display 기술을 바탕으로 보다 그래픽한 정보를 제공할뿐더러, SOC (System On a Chip) 기술의 도입을 통하여 PC에서의 컴퓨팅 기능을 Portable Device에서 제공할 수 있을 정도로 그 응용영역의 폭이 넓어지고 있다. 이에 따라 휴대용 게임기, 차세대 PDA 및 핸드폰 등에 3차원 영상지원이 요구되어지고 있다. 하지만 기존의 3차원영상 처리기는 저전력, 소규모를 요구하는 휴대형 기기에는 적합하지 않다. 따라서 본 논문에서는 휴대형 기기에 적합한 3차원 영상처리기를 제안한다[1,2,6].

Ⅱ장에서는 일반적인3차원 그래픽 처리 과정과 본 논문에서 제안하는 모바일 기기에 적합한 내장형 3차원 그래픽 렌더링처리기에 대하여 설명한다. Ⅲ장에서는 제안한 내장형 3차원 그래픽 렌더링 처리기의 성능을 분석 하고 Ⅳ장에서 결론을 맺는다.

* 정회원, (주) 코아로직
(Corelogic)
** 학생회원, *** 정회원, 연세대학교 전기전자공학과
(Dept. of Electrical Electronic Engineering, Yonsei University)
접수일자: 2004년4월2일, 수정완료일: 2004년10월30일

Ⅱ. 3차원 그래픽 처리과정

1. 3차원 그래픽 처리과정

3차원 그래픽 가속기는 실시간으로 사용자의 입력을 반영해 영상 데이터를 변형, 가공하는 장치로 3차원 영상 데이터의 크기 및 위치 변화와 그에 따른 빛의 변화를 처리하는 기하학 연산 처리기(Geometry Processor)와 영상 데이터의 색상 및 텍스쳐를 입히는 렌더링 처리기(Rendering Processor)로 이루어진다. 일반적인 PC와 같은 고성능의 컴퓨팅 환경에서는 기하학 처리 부분과 렌더링 처리 부분을 모두 하드웨어로 구현하고 있다[10,11]. 그러나 모바일 환경에서 이 두 부분을 모두 하드웨어로 구성하는 것은 무리가 있으며 렌더링 처리기만을 하드웨어로 구성하여도 충분한 성능을 얻을 수 있다. 따라서 이 논문에서는 위의 가정을 바탕으로 렌더링 처리 부분만을 하드웨어로 구현하였으며 이에 대한 자세한 논의 및 성능 분석은 제3장에서 하겠다.

가. 기하학 연산처리기 (Geometry Processor)

한 화면을 이루는 영상은 여러 개의 객체가 모여 있으며 현실감을 높이기 위해 한 화면을 구성하는 객체의 수는 갈수록 증가하는 추세이다. 이러한 객체들을 한 화면에 표현하기 위해서는 모든 객체들을 통합된 하나의 공간 안에 존재하도록 해야 한다. World Transform은 각 객체를 이루는 삼각형들의 정점의 좌표를 객체 좌표계에서 통합 좌표계로 변환하여 모든 삼각형들이, 즉 모든 객체들이 하나의 공간 안에 존재하도록 한다. World Transform을 포함한 이후의 모든 변환은 벡터 행렬 곱셈으로 이루어진다.

World Transform을 한 삼각형은 Lighting 과정을 거치게 된다. 현실 세계는 여러 개의 빛이 존재하여 그 빛의 밝기와 색이 객체의 색상에 영향을 주며 객체 표면의 빛의 반사도 객체의 색상에 영향을 준다. Lighting 과정은 이러한 빛에 의한 삼각형의 색상 변화를 연산한다[8].

Lighting 과정 후에 통합 좌표계를 시점 좌표계로 변환하는 View Transform을 하는데, View Transform은 카메라와 같은 시점의 위치와 방향을 결정하여 모든 삼각형들을 그 시점에 의한 시점 좌표계로 표현하는 것이다. 시점의 위치와 방향에 따라 삼각형을 옮기거나 회전시키거나 삼각형의 크기를 변화시킨다. 사용자의 시점은 실시간으로 무작위 방향으로 변화하며 현실감 있는 영상을 위해 다 시점 처리도 도입되고 있다.

지금까지는 가상적인 3차원 공간 내에 삼각형들이 존재하였지만 3차원 영상 데이터를 2차원 화면에 표현하기 위해서 이 삼각형들을 2차원 평면에 투영시키는 과정이 필요하다. 이를 Projection Transform이라고 한다. 2차원 평면에 투영시킬 때는 가상적인 Z축으로 시점과 삼각형 사이의 거리를 나타내어 삼각형간의 앞뒤를 표현한다. 삼각형의 Z값은 렌더링 처리기에서 깊이 비교를 수행할 때 사용된다. 마지막으로 Clipping 과정에서는 실제로 화면에 보여 지는 영역인 View Volume 밖에 있는 삼각형들을 잘라내어 불필요한 연산을 줄여준다.

이런 과정을 거쳐 기하학 연산 처리기는 각 객체 공간 속에 있는 삼각형들을 하나의 2차원 화면에 표현하고, 각 삼각형의 정점의 좌표와 색상, 가상적인 Z값을 연산한다. 기하학 연산 처리가 끝난 삼각형 데이터는 렌더링 처리기로 보내게 된다. 현실감 있는 영상을 표현하기 위해 영상이 보다 정밀해지면서 한 화면을 구성하는 삼각형의 수가 많아지고 시점도 매순간 변하게 되어 기하학 연산 처리기는 점점 더 성능의 향상이 요구되고 있다.

나. 렌더링 처리기 (Rendering Processor)

렌더링 처리기에서는 기하학 연산 처리가 끝난 후 삼각형 내부의 색상을 결정해 주는 부분으로 현실감 있는 영상을 제공하기 위해 텍스쳐 매핑과 투명도 처리, 깊이 비교, 안개 처리, 안티앨리어싱 등의 기법 연산도 수행한다[10,11].

기하학 연산 처리가 삼각형 단위로 이루어졌다고 하지만 엄밀히 말하면 삼각형의 세 정점에 대해 이루어진 것이므로 우선 이 세 정점을 가지고 가상적인 삼각형을 만들어야 한다. 삼각형 셋업(Triangle Setup)은 이 세 정점을 가지고 삼각형을 이루는 요소인 각 변의 기울기와 색상 정보의 증가분을 구하는 과정으로 가상적인 삼각형을 만들기 때문에 삼각형 셋업이라고 한다. 삼각형 셋업을 한 후 각 변의 기울기와 색상 정보의 증가분을 이용하여 각 변을 이루는 픽셀의 색상 정보를 구하게 되는데 이를 변 처리(Edge Processing)라고 한다. 변 처리 과정을 거친 후, 마지막으로 각 주사선에 있는 모든 픽셀에 대해서 색상 정보를 구하는 것을 스팬 처리(Span Processing)라고 한다. 폴리곤 처리, 변 처리 및 스팬 처리를 주사 변환(Scan Conversion) 과정이라고도 하며, 이와 같은 과정을 통하여 삼각형 내부의 픽셀들에 대한 색상 정보를 구하고 이를 메모리에 저장한다.

현실감 있는 영상을 제공하기 위해서 렌더링 처리기에서는 주사 변환 과정 이후에 추가적인 영상 처리 과정(Realization)을 수행한다. 텍스쳐 매핑, 투명도 처리, 안개 처리, 깊이 비교 등의 과정이 렌더링 처리기의 후반부에서 이루어진다.

화면에 보여 지는 객체가 단순한 색만을 가지지 않고 어떤 이미지를 가지게 되면 영상이 보다 자연스러울 것이다. 텍스쳐 매핑은 이미지를 화면 내의 객체의 표면에 투영하는 방법으로 텍스쳐 데이터를 각 픽셀에 매핑하는 것이다. 시점과의 거리에 따라 객체의 크기가 변하므로 이에 따라 텍스쳐 데이터의 크기도 변해야 한다. 그래서 동일한 이미지에 대하여 다양한 크기의 텍스쳐 데이터를 제공하는 텍스쳐 밉매핑(Texture Mip-mapping) 방식이 주로 사용된다[10].

2. 제안하는 내장형 3차원 그래픽 렌더링 처리기

제안하는 내장형 3차원 그래픽 렌더링 처리기는 고라우드 쉐이딩(Gouraud Shading)을 사용하였고, 800*600의 해상도와 r, g, b, a값이 각각 8비트인 True 칼라 및 16비트의 깊이정보를 지원하도록 설계하였다.

제안하는 3차원 그래픽 렌더링 처리기는 소규모 구현을 위하여 고정 소수점 연산을 사용하였으며, 반복적인 구조로 동일 연산 및 유사 연산을 수행하는 부분을 반복 사용함으로 하드웨어의 크기를 줄일 수 있었다. 하지만 이러한 반복적인 연산이 하드웨어의 성능에 큰 영향을 주어서는 안 될 것이다. 따라서 각 처리 단이 독립적일 수 있도록 변 처리를 에지 워크(Edge Walk)와 스캔라인 처리로 나누었으며, 삼각형 당 1회 수행되는 삼각형 셋업과 2회 수행하는 에지워크(Edge Walk)는 반복구조로 삼각형의 라인 수만큼 수행하는 스캔라인 처리 및 스팬의 픽셀 수만큼 수행하는 스팬 처리는 반복구조를 사용하지 않았다. 또 기존 삼각형 셋업에서 병렬적으로 사용되는 7개의 나눗셈기를 하나의 나눗셈기로 재귀적인 사용을 함으로 하드웨어의 크기가 큰 나눗셈의 문제를 해결할 수 있었다. 사용한 나눗셈기는 5단 파이프라인을 가진 테일러시리즈 멀티플리케이티드 방식으로 결과 값이 최하위 비트(LSB)에서 한 비트의 오차를 갖지만 시뮬레이션을 통하여 그 오차가 무시 될 수 있음을 확인하였다.

그림 1은 설계한 내장형 3차원 그래픽 렌더링 처리기가 포함 된 SoC이다. 기하학 연산 처리는 ARM10과 VFP10(부동소수점 연산기)가 소프트웨어로 처리하며, 처리된 데이터는 버텍스 버퍼에 쌓이게 되고 커맨드 파서를 이용 32비트씩 읽어 V0,V1,V2 3개의 정점에 대한 정보를 삼각형 처리기로 보내 처리 하게 된다. 그림에서 점선 안의 부분이 하드웨어로 설계된 부분이고 그 중 본 논문에서는 텍스쳐 매핑까지의 구현을 제안한다.

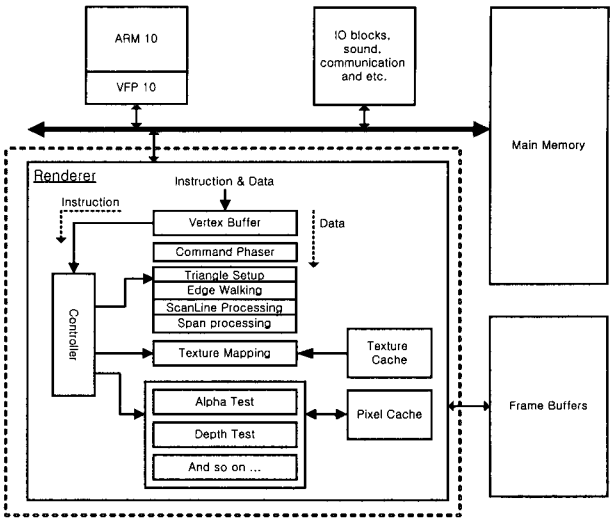


그림 1. 제안하는 3차원 그래픽 렌더링 처리기 SoC
Fig. 1. Proposed 3D graphics rendering processor.

가. 렌더링 처리 하드웨어의 필요성

Ⅱ장 1절에서 언급한 바와 같이 3차원 그래픽 가속기를 전부 하드웨어로 구현하기에는 모바일 기기에서는 그 부담이 크며, 또 소프트웨어로 처리해서는 원하는 성능을 얻을 수 없다. 표 1과 표 2는 메사 라이브러리를 ARM10과 VFP10(부동소수점 연산기)을 이용하여 소프트웨어로 처리한 결과이다. 실험 데이터로는 메사 라이브러리에서 지원하는 Cube와 Flight simulation을 사용하였다. Total Frame 처리시간은 한 프레임을 모두 소프트웨어로 처리한 결과이며 프레임 처리시간이 해상도에 따라 기하급수적으로 증가함을 알 수 있다. 이에 반해 Only Geometry Frame 처리시간은 기하학 연산처리만 소프트웨어로 처리하였을 때에 한 프레임을 처리하는 시간이다. 즉 소프트웨어로만 처리하면 Cube일 경우 초당 2~14 프레임을 처리하고 Flight일 경우 한 프레임을 처리하는 데 3~21초가 소요된다. 이는 모바일 기기일지라도 원하는 성능을 내기에는 턱없이 부족하다. 하지만 기하학 연산처리만을 소프트웨어로 할 경우 Cube는 초당 31~33 프레임을, Flight는 초당 13~20 프레임을 처리한다. 따라서 제안하는 렌더링 처리기는 Cache Miss 등을 고려 30 frame/sec이상의 성능을 목표로 하였다[12].

표 1. Cube 시뮬레이션 결과
Table 1. Cube simulation.

해상도	Total Frame 처리시간 (초)	Only Geometry Frame 처리시간 (초)	프레임 당 픽셀 수
320*240	0.068733	0.026564	41972 pixel
640*480	0.277719	0.026773	167768 pixel
800*600	0.462318	0.031333	262244 pixel

표 2. Flight 시뮬레이션 결과
Table 2. Flight simulation.

해상도	Total Frame 처리시간 (초)	Only Geometry Frame 처리시간 (초)	프레임 당 픽셀 수
320*240	3.395301	0.048923	330428 pixel
640*480	12.541485	0.063440	1319010 pixel
800*600	21.019301	0.076004	2062972 pixel

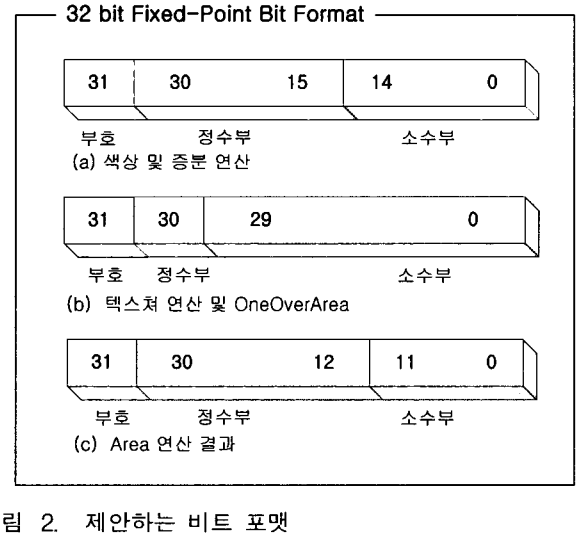


그림 2. 제안하는 비트 포맷
Fig. 2. Proposed bit format.

나. 고정 소수점(Fixed point) 연산의 비트 포맷

제안하는 3차원 그래픽 렌더링 처리기는 모바일의 요구사항인 소규모를 위해 기존의 Single precision Floating-Point 연산 대신 고정소수점 연산을 사용하였다.

사용한 고정소수점 연산 포맷은 삼각형의 색상 및 좌표 그리고 증분연산을 위한 포맷(a), 0~1사이의 값만을 갖는 소수점에 민감한 텍스쳐 연산 및 OneOverArea 연산 포맷(b), Area연산 포맷(c)으로 나누어진다[그림 2].

고라우드 쉐이딩 환경에서는 고정소수점 연산으로 야기되는 증분 값의 오차가 각 픽셀별로 누적이 되기 때문에 오버플로우와 언더플로우를 고려하여 비트의 포맷을 신중히 결정해야 할 것이다. 그러나 만약 고정소수점 연산의 비트 포맷이 다양할 경우 고정소수점 연산을 사용하여 얻는 이점을 잃을 뿐만 아니라 그 구조도

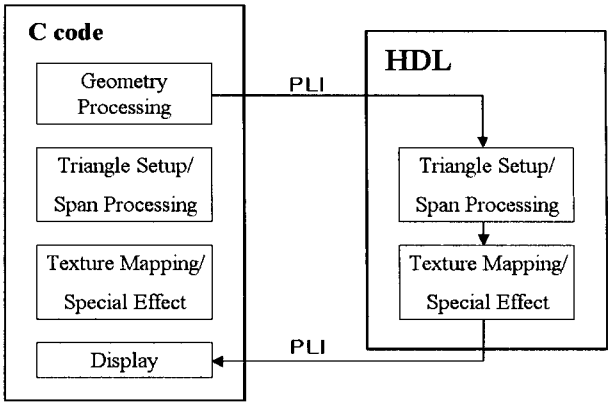


그림 3. 실험환경
Fig. 3. Simulation environment.

복잡해 질것이다. 따라서 RTL 설계 전 동작수준에서의 다양한 시뮬레이션으로 그림 2와 같은 비트 포맷을 얻을 수 있었다.

오버플로우의 발생은 영상 결과에 치명 적이다. 따라서 오버플로우가 일어날 수 있는 모든 가능성을 배제하여야 한다. 오버플로우가 발생할 수 있는 대표적인 부분으로 삼각형의 넓이, 그리고 증분 연산 등의 곱셈 결과에서 들 수 있다. 언더플로의의 경우 그 해결 방법이 단지 정확한 비트의 확장만이 가능하지만, 오버플로우의 경우는 단순히 비트를 확장하는 것에 끝나지 않고 두 곱셈이 큰 값 * 작은 값을 취하게 하여 해결 할 수 있었다.

$$\frac{dC}{dy} = OneOverArea^{*}\,(eMaj.dx^{*}\,eBot.dC - eMaj.dC^{*}\,eBot.dx) \quad (1)$$

$$\frac{dC}{dy} = (OneOverArea^{*}\,eMaj.dx)^{*}\,eBot.dC - (OneOverArea^{*}\,eMaj.dC)^{*}\,eBot.dx \quad (2)$$

위의 식 (1)에서 곱셈의 순서를 바꾸어주어 오버플로우를 해결할 수 있는데 OneOverArea는 삼각형의 크기에 반비례하고, eMaj.dx, eBot.dx는 비례하기 때문에 최악의 경우(밑변 800, 색상 차 255) " eMaj.dx * eBot.dC = 800*255 = 204000" 즉, 18비트가 필요하게 되지만 연산 순서를 바꾸어 주게 되면 "Area = 600*800/2 = 2400, OneOverArea = 1/2400, (800*1/2400)*255 = 85" 이 된다. 이러한 방법으로 오버플로우의 발생 가능성을 배제할 수 있었다.

동작수준 테스트 모델은 그림 3과 같으며 제안한 아키텍처의 검증을 위하여 메사 라이브러리를 수정한 C 코드와 제안하는 하드웨어와의 PLI를 통하여 화면에

최종 디스플레이 하는 방식을 사용하였다. 실험 데이터로는 메사 라이브러리에서 제공하는 DYN과 Cube와 Flight simulation을 사용하였고, 다양한 경우를 고려하여 검증을 하였다[9,12].

다. 명령어 처리기(Command Parser)

그래픽프로세서에 입력된 명령어와 데이터 처리를 위해 Command Parser를 사용한다. Command Parser는 Vertex Buffer라고 불리는 일종의 명령어와 데이터의 묶음을 받아서 명령어를 분석하고 데이터를 정렬하여 하단의 Rasterizer에 필요한 데이터와 제어 신호를 보낸다. Command Parser의 동작은 그림 4와 같다. 우선, 명령어와 데이터 셋인 Vertex Buffer가 들어올 때까지 대기 상태(INIT)로 있다가 Vertex Buffer가 들어오면 그 중 Header 부분을 분석한다. Vertex Buffer의 구조는 그림 5와 같다. Header 부분에는 Rasterizer 및 그 하단 부분에서 필요한 다양한 정보들을 포함하고 있다. Header 부분에 대한 분석이 끝나면 Command Parser는 삼각형을 처리하기 위한 동작(Tri Index)으로 넘어가게 된다. 그래픽 데이터를 처리하기 위한 기본

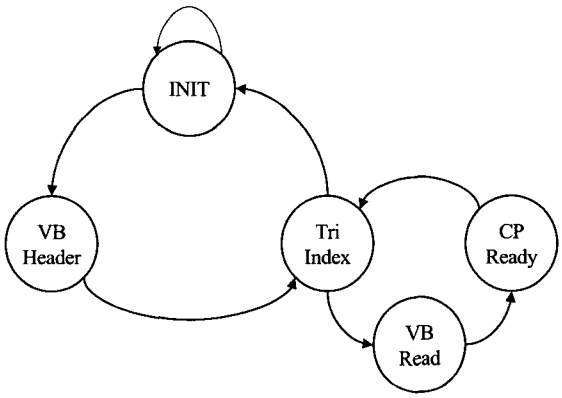


그림 4. 커맨드 파서의 동작
Fig. 4. Command parser.

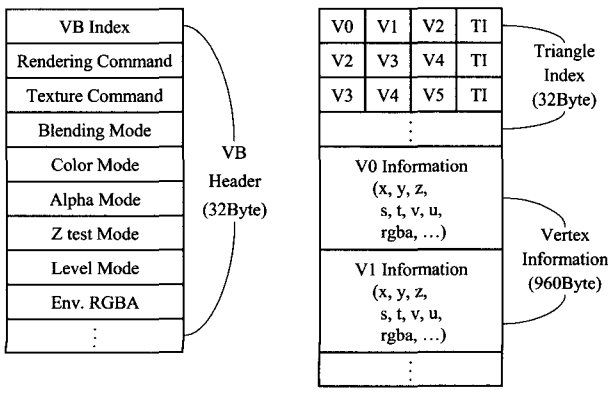


그림 5. Vertex Buffer의 구조
Fig. 5. Vertex buffer format.

단위인 폴리곤은 대부분 삼각형으로 구성되어 있다. 삼각형을 구성하는 세 정점과 그 삼각형을 처리하는 데 필요한 정보를 가지고 있는 부분이 Vertex Buffer에서 Triangle Index이다. V0, V1, V2, V3는 각각 삼각형을 이루는 정점을 나타내며 TI(Triangle Information)는 Texture Index, Frame Information 등 삼각형에 대한 처리 정보이다. 삼각형에 대한 세 정점을 알게 되면 그 정점에 해당하는 좌표 정보, 색상 정보, 텍스쳐 정보 등을 Vertex Buffer의 Vertex Information에서 가져오게 된다. 세 정점에 대한 정보를 모두 가져온 후에는 하단에서 요구가 있을 때까지 대기(CP Ready)하게 된다. 정점에 대한 Index와 정보를 따로 관리하는 것은 정점이 여러 삼각형에 중복되어 사용되기 때문에 재사용의 편의성을 위한 것이다. 하단의 요구로 정점을 포함한 삼각형에 대한 모든 정보를 보낸 후에는 Tri Index 상태로 되돌아와 앞의 동작을 반복하게 된다. Vertex Buffer에 있는 데이터를 모두 처리하게 되는 경우는 TI에 있는 Frame Information에 이미 표시를 해주게 되고 이를 보고 판단하여 새로운 Vertex Buffer를 불어오게 된다. Frame이 끝나는 경우도 유사한 방법으로 처리하게 된다.

라. 삼각형 셋업, 에지 워크, 스캔 라인, 스팬 처리기

제안하는 3차원 그래픽 렌더링 처리기는 소규모 구현을 위하여 반복적인 구조로 동일 연산 및 유사 연산을 수행하는 부분을 반복 사용함으로서 하드웨어의 크기를 줄일 수 있었다. 하지만 이러한 반복적인 연산이 하드웨어의 성능에 큰 영향을 주어서는 안 될 것이다. 따라서 각 처리 단이 독립적일 수 있도록 변 처리를 에지 워크(Edge Walk)와 스캔라인 처리로 나누었으며, 삼각형 당 1회 수행되는 삼각형 셋업과 2회 수행하는 에지 워크(Edge Walk)는 반복구조로 삼각형의 라인 수만큼 수행하는 스캔라인 처리 및 스팬의 픽셀 수만큼 수행하는 스팬 처리는 반복구조를 사용하지 않았다. 또 기존 삼각형 셋업에서 병렬적으로 사용되는 7번의 나눗셈 연산을 처리율이 1 cycle인 파이프라인 나눗셈기를 사용하여 하나의 나눗셈기로 재귀적인 사용을 함으로 하드웨어의 크기가 큰 나눗셈의 문제를 해결할 수 있었다. 사용한 나눗셈기는 5단 파이프라인을 가진 테일러 시리즈 멀티플리케이티드 방식으로 결과 값이 최하위 비트(LSB)에서 한 비트의 오차를 갖지만 시뮬레이션을 통하여 그 오차가 무시 될 수 있음을 확인하였다[6,7].

표 3은 3차원 삼각형 셋업, 에지 워크, 스캔라인, 스

표 3. 필요한 연산수와 지연시간
Table 3. Arithmetic unit and latency.

		필요 연산의 수	유닛수(Ver3.0)
삼각형 셋업	곱셈	65	7
	나눗셈	7	1
	Latency		34
변 처리	곱셈	31	2
	나눗셈	0	0
	Latency		17
스캔라인/ 스팬처리	곱셈	7	7
	나눗셈	1	1
	Latency		1
Total	곱셈	103	16
	나눗셈	8	2
	삼각형 당 추가지연	0 cycle	68 cycle

팬 처리 과정에서의 필요 연산수와 제안하는 반복 구조에서의 연산기수 및 그에 따른 추가 latency 이다.

마. Texture Mapping

3차원 영상 데이터에 현실감을 더해주는 과정이 Texture Mapping 처리이다. 최근에는 Lighting 처리, Bump Mapping, Environment Mapping까지 Texture Mapping으로 처리하고 있으며 한 픽셀에 최대 8개까지의 Texture Mapping을 처리한다.

Texture Map을 입히는 과정에서 원근에 따라 영상이 왜곡되는 Rubber Sheet 현상이 발생한다. 이를 막기 위해서 식 (3)과 같은 Perspective Division을 통해 원근을 처리해주는 하드웨어를 추가하였다[9].

$$\frac{ds}{dx} = \frac{d(S/W)}{dx} = \frac{(dS/dx)*W - (dW/dx)*S}{W^2} \quad (3)$$

폴리곤의 크기에 따라 Texture Map의 크기를 바꾸어주는 Bi-linear Mip-mapping을 구현하였다. Mip-map의 레벨을 결정하는 다양한 알고리즘[15]이 있는데 식 (4)와 같은 수식을 이용하여 하드웨어의 사용을 최소화하였다.

$$L = log_2 max\left(\frac{ds}{dx}, \frac{dt}{dx}, \frac{ds}{dy}, \frac{dt}{dy}\right) \quad (4)$$

Bi-linear 보간 과정에서도 Gaussian 함수[16]를 근사화한 수식을 사용하여 영상의 질을 높이기 위해 노력하였다. 그림 6은 보간 과정에서 Linear 함수, Gaussian 함수, 제안한 함수를 사용하였을 때의 입출력 값의 변화이다. Direct3D와 OpenGL에서 요구하는 6가지의 블렌딩 모드와 5가지의 칼라 모드를 모두 지원한다. 또한 이 과정에서 효율적으로 Texture Map 데이터를 가져오기 위해 Texture Cache를 구현하여 사용하였다.

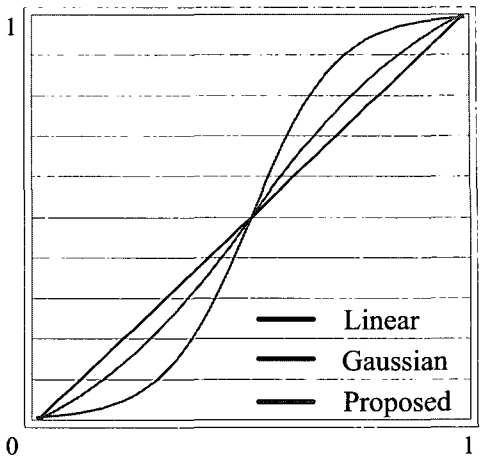


그림 6. 보간 과정상의 입출력 변화
Fig. 6. Bi-linear interpolation.

Ⅲ. 실 험

1. FPGA 및 ASIC공정 합성

제안하는 반복적 연산을 하는 내장형 3차원 그래픽 렌더링 처리기는 800*600의 해상도와 트루컬러를 지원하도록 설계하였으며 하드웨어의 성능 향상 및 FPGA 보드 검증을 용이하게 하기 위하여 플립플롭 기반으로 설계하였다. Xilinx Virtex2 6000을 사용하여 검증을 완료하였고 동작 주파수는 36.4MHz 이었으며 LUT의 55%를 차지하였다.

Virtex2 6000은 144개의 18*18 곱셈기가 내장 되어 있어서 32*32 곱셈기는 4개의 18*18유닛으로 합성이 되며, 하나의 나눗셈기에는 4개의 32*32 곱셈이 있어 이는 16개의 18*18곱셈기로 합성이 된다. 전체 구조에서 필요한 곱셈기가 Virtex2 6000에 내장된 개수를 초과하게 되어 Core generator를 이용하여 부족한 곱셈기를 LUT 합성하여 사용하였다. 그림 7은 FPGA 검증 시 사용한 보드이다.

Synopsys툴을 사용하여 0.25um ASIC 공정에서 합성하였고 50MHz의 동작 주파수와, 64만 게이트 사이즈 얻을 수 있었다. 그림 8은 Apollo를 사용한 레이아웃의 결과이다. 면적은 4.9287mm*4.9847mm이고 629,000*4개의 트랜지스터로 구성되며, 50MHz의 동작주파수를 가지며, 파워소모는 263.7mW이다.

그림 9는 0.25um ASIC 공정상의 유닛별 하드웨어

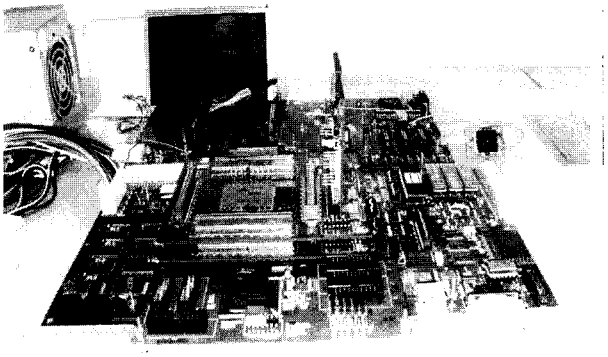


그림 7. FPGA 보드
Fig. 7. FPGA.

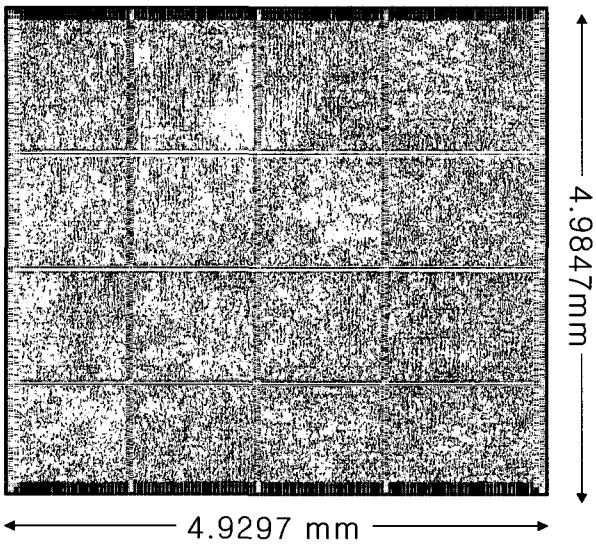


그림 8. 제안하는 렌더링 처리기 레이아웃
Fig. 8. Layout.

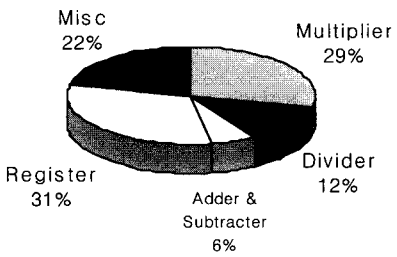


그림 9. 렌더링 처리기의 유닛별 비율
Fig. 9. Area.

비율을 나타낸 것으로 레지스터가 198,984 게이트, 곱셈기가 185,110 게이트 나눗셈기가 79,658 게이트, 덧셈 및 뺄셈기가 38003 게이트, 먹스 및 버퍼 등이 138245 게이트를 차지하였다.

표 4. Z3D와의 비교
Table 4. Z3D vs. Proposed rendering processor.

	Z3D	제안하는 하드웨어
면적	36만 게이트 (11mm)	64만 게이트 (24.56mm)
공정	0.18um	0.25um
동작주파수	30Mhz	50Mhz
초당 픽셀수	5.1M Pixel/sec	47.88M Pixel
초당 프레임 처리	19.1 frame /sec	33.25 frame /sec
해상도	120*120(추정)	800*600
지원칼러	24bit	32bit

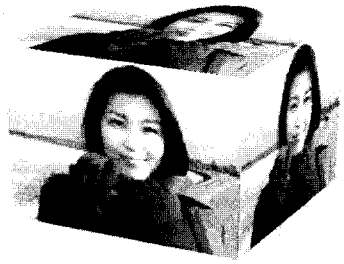


그림 10. Cube
Fig. 10. Cube.

2. 성능 분석 및 비교

제안하는 반복구조에서는 삼각형 당 68사이클의 Latency를 가진다. 따라서 ASIC공정에서 50MHz로 동작 시킨다면 일반적인 테스트 데이터인 퀘이크 3의 경우 데모 1의 경우 45.7M pixel/s의 성능을 보이며, 데모 2의 경우 47.88M pixel/s의 성능을 보인다. 초당 프레임 수는 초당 픽셀수를 프레임들의 평균 픽셀수로 나누면 구할 수 있으며 최대 33.5 frame/sec의 처리율을 갖는다.

현재 많은 업체들이 모바일용 3D 그래픽 가속기를 연구개발 중에 있으며, ARM은 이매지네이션과 제휴 파워VR 코어를 기반으로 한 통합 그래픽 프로세싱 코어인 MBX IP를, Mitsubishi는 핸드폰 전용인 Z3D IP를, 한국의 Magic eyes사는 휴대용 3D게임기, 3D 게임폰을 위한 Vender 3D IP 등을 출시하고 있으나, 그 내부의 스펙은 공개하지 않고 있다. 그 중 스펙이 공개된 모바일 기기에 관한 IP인 Z3D를 비교 대상으로 삼았다[14]. 표 4는 Z3D와 제안하는 3차원 영상 처리기의 성능을 나타낸 것으로 지원하는 사양이 다르기 때문에 Z3D

그림 11. DYN
Fig. 11. DYN.

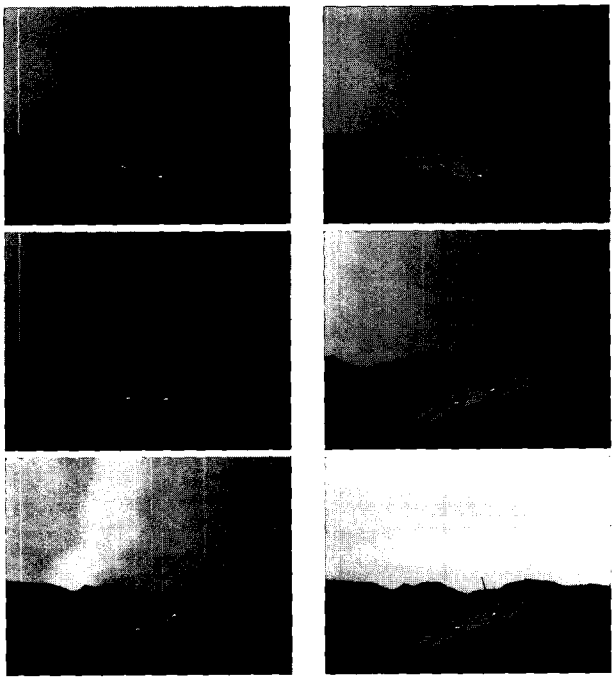


그림 12. Flight
Fig. 12. Flight.

와 직접적인 비교를 하기는 어렵다. 단지, 본 논문에서 제안한 3차원 그래픽 렌더링 처리기가 800×600의 고해상도에 트루 컬러를 지원하는 것에 비해 면적은 약 2배 정도만 크다. 구현한 3차원 그래픽 렌더링 처리기의 사양을 낮출 경우, Z3D보다 작은 면적으로 설계할 수 있을 것으로 기대된다.

Ⅳ. 결 론

휴대형 기기의 대중화에 따라 3D 영상처리 하드웨어의 필요성은 증가 할 것이다. 이에 본 논문은 휴대형 기기에 적합한 3차원 렌더링 처리기를 설계하였다. 설계한 하드웨어는 800*600의 고해상도 및 고급 색품질을 지원하며 PDA등의 고사양 휴대형 기기에 적합하며, 0.25um ASIC 공정에서 합성한 결과 50MHz의 동작 주파수에서 63만 게이트를 차지하며 4.9847mm × 4.9287mm 의 크기와 263.7mW의 전력을 소모한다[그림 8].

참 고 문 헌

[1] Hyun-Jea Woo, Jong-Chul Jeong, Moon-Key Lee:"The Design of efficient 3D graphics Rende -ring Architecture for mobile device "Proc of KGS pp337~342, Jan,20

[2] Hyun-Jea Woo, Jong-Chul Jeong, Moon-Key Lee. "An efficient, pipelined architecture for 3D graphics accelerator", Proc. of IEEK Summer Conference 2003, No.2, pp.1213-1216, July, 2003, Kor

[3] Hyun-Jea Woo, Jong-Chul Jeong, Moon-Key Lee:"The Design of 3D Graphics Rendering Processor for Portable Devices "Proc of 2003 SOC Design Conference ,NOV,20

[4] Hyun-Jae Woo, Jong-Chul Jeong, Moon-Key Lee, "An Efficient Pipelined Architecture for 3D Graphics Accelerator", Proc. of IEEK Summer Conference 2002, No.2, pp.357-360, June, 2002, Korea

[5] Jong-Chul Jeong, Woo-Chan Park, Moon-Key Lee, Tack-Don Han, "A New Technique to Solve Consistency Problem for 3D Parallel Raste -rizer on a Single Chip", Proc. of Chunchon Multimedia Conference, pp.447-450, June, 2001, Chunchon, Korea

[6] Woong Jeong, Woo-Chan Park, Sung-Ho Kwak, Hoon-Mo Yang, Cheol-Ho Jeong, Tack-Don Han, and Moon-Key Lee, "A New Pipelined Divider with a Small Lookup Table", Journal of IEEK, Vol. 40, No.9 pp.94-104, Sep. 2003-09-12

[7] Jong-Chul Jeong, Woong Jeong, Hyun-Jae Woo, Seung-Ho Kwak, Woo-Chan Park, Moon-Key Lee, Tak-don Han, "A New Pipelined Divider with a Small Lookup Table", Proc. Of 2002 IEEE ASIA-PACIFIC Confernence on AISC, pp.33-36, August 6-8, 2002, Taipei, Taiw

[8] D. Kirk, Unsolved Problems and Opportunities for High-Quality, High-Performance 3-D Gra -phics on a PC Platform, Proceedings of SIGGRAPH /Eurographics Workshop on Graphics Hardware, Keynote talk, Aug. 1998.

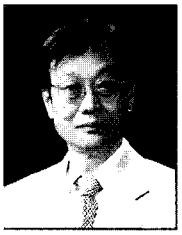
[9] A. Kugler, The Setup for Triangle Rasterization,

11th Eurographics Workshop on Computer Gra -phics Hardware, pp. 1-10, August 1996.
[10] Intel, Intel740 Graphics Accelerator Software De -veloper's Manual, Feb. 1998, http://support.intel.com/support/graphics/intel740/.
[11] J. McCormack, R. McNamara, C. Cianos,N. P. Jouppi, T. Dutton, J. Zurawski, L. Seiler, K. Corell, Implementing Neon : A 256-bit Graphics Accelerator, IEEE Micro, pp. 58-69, April 1999.
[12] Mesa library, http://www.ssec.wisc.edu/~brianp/Mesa.html.
[13] Quake 3, http://www.idsoftware.com/games/quake/quake3-aren
[14] M Kameyama, "3D Graphics LSI Core for Mobile Phone "Z3D", Graphics Hardware 2003, pp.60-67
[15] Jon P. Ewins, Marcus D. Waller, Martin White and Paul F. Lister, "Mip-Map Level Selection for Texture Mapping," IEEE Transaction on Visualization and Computer Graphics, Vol.4, No.4, Oct.-Dec. 1998.
[16] Compaq, "Neon: A (Big) (Fast) Single-Chip 3D Workstation Graphics Accelerator," July, 1999.

저 자 소 개

우 현 재(정회원)
2002년 명지대학교 전자공학과 학사 졸업
2004년 연세대학교 전기전자공학과 석사졸업
<주관심분야: 3차원 그래픽 가속기, 반도체, 마이크로프로세서>

장 태 홍(학생회원)
2004년 연세대학교 기계전자공학부 전기전자전공 학사 졸업
2004년~현재 연세대학교 전자공학과 석사 과정
<주관심분야: 3차원 그래픽 가속기, 마이크로프로세서, 저전력 설계>



이 문 기(정회원)
1965년 연세대학교 전기공학과 학사 졸업
1967년 연세대학교 전자공학과 석사 졸업
1973년 연세대학교 전자공학과 박사 졸업
1980년 UNIV. of OKLAHOMA 전자공학 박사 졸업
1982년~현재 연세대학교 전기전자공학과 교수
<주관심분야: 마이크로프로세서, 반도체>