논문 2006-43SD-2-5

내장형 3D 그래픽 가속을 위한 부동소수점 Geometry 프로세서 설계

(A Design of Floating-Point Geometry Processor for Embedded 3D Graphics Acceleration)

남 기 훈****, 하 진 석*, 곽 재 창****, 이 광 엽**

(Ki hun Nam, Jin Seok Ha, Jae Chang Kwak, and Kwang Youb Lee)

요 약

본 논문에서는 휴대용 정보기기 시스템에서 더욱 향상된 실시간 3D 그래픽 가속 능력을 갖는 SoC 구현을 위해 효과적인 3D 그래픽 Geometry 처리 IP 구조를 연구하였다. 이를 기반으로 3D 그래픽 Geometry 처리 과정에 필요한 부동소수점 연산기를 설계하였으며, 내장형 3D 그래픽 국제 표준인 OpenGL-ES를 지원하는 부동소수점 Geometry 프로세서를 설계하였다. 설계된 Geometry 프로세서는 Xilinx-Vertex2 FPGA에서 160K gate의 면적으로 구현되었으며, 80 MHz의 동작주파수 환경에서 실제 3D 그래픽 데이터를 이용하여 Geometry 처리 과정의 성능 측정 실험을 하였다. 실험 결과 80 MHz의 동작주파수에서 초당 1.5M개의 폴리곤 처리 성능이 확인되었으며, 이는 타 3D 그래픽 가속 프로세서에 비하여 평균 2배 이상의 Geometry 처리 성능이다. 본 지오메트리 프로세서는 Hynix 0.25um CMOS 공정에 의한 측정결과 83.6mW의 소모전력을 나타낸다.

Abstract

The effective geometry processing IP architecture for mobile SoC that has real time 3D graphics acceleration performance in mobile information system is proposed. Base on the proposed IP architecture, we design the floating point arithmetic unit needed in geometry process and the floating point geometry processor supporting the 3D graphic international standard OpenGL-ES. The geometry processor is implemented by 160K gate area in a Xilinx-Vertex FPGA, and we measure the performance of geometry processor using the actual 3D graphic data at 80MHz frequency environment. The experiment result shows 1.5M polygons/sec processing performance. The power consumption is measured to 83.6mW at Hynix 0.25um CMOS@50MHz.

Keywords: 3D Graphics accelerator, Floating-Point Unit, Geometry processor

I. 서 론

최근 휴대 통신 회사는 무선 통신을 통해 3D 그래픽 아바타 모델 서비스와 3D 그래픽 게임 서비스를 하기 시작하였다. 하지만 이러한 3D 그래픽 기반의 서비스는 현재 휴대 정보 기기의 처리 속도 및 전력 소비 등 기술적인 한계가 있다. 이러한 기술적인 한계를 해결하기 위해 현재는 그림 1의 3D 그래픽 라이브러리 계층 중 type-1과 같이 CPU 프로그래밍 기반으로 3D 그래픽을 가속하고 있다[1]. 그러나 하드웨어 가속이 지원되지 않는 모바일 환경에서 최대 5~6만 polygon/sec 이상의 성능을 기대하기 어려우며, 이는 곧 3D 그래픽의 강점이라 할 수 있는 입체감과 사실감을 표현 해 주는 높은

* 학생회원, ** 정회원, 서경대학교 컴퓨터공학과
(Dept. of Computer Engineering SeoKyeong Univ.)
*** 학생회원, **** 정회원, 서경대학교 컴퓨터과학과
(Dept. of Computer Science SeoKyeong Univ.)
※ 본연구는 ITSOC 사업단과 서울시 혁신 클러스터 육성사업의 지원으로 수행 되었으며, IDEC 지원 장비를 활용하였습니다.
접수일자 : 2005년4월28일 수정완료일 : 2006년1월16일

그래픽 품질이나 게임에 있어 중요한 요소인 속도감 그리고 흥미를 배가시키는 정교한 게임의 구현이 어려움을 의미한다. 더욱 향상된 3D 그래픽 처리 능력 위해 정점의 좌표와 광원에 대한 정점의 색을 정하는 Geometry 단계와 픽셀의 색과 좌표를 정하는 Rendering 단계를 하드웨어로 가속하는 type-2, type-3과 같은 구조가 필요하다. 그러나 type-2는 Geometry 단계를 CPU 프로그래밍에 의존하고, Rendering 단계를 하드웨어 가속하는 구조이다. 이러한 구조는 90년 후반부터 현재까지 PC의 3D 그래픽 가속에서 사용되는 구조로 nVidia의 Geforce 시리즈와 ATI의 Radeon 시리즈의 다수의 3D 그래픽 가속 카드가 이에 속한다. 그러나 PC의 CPU와 달리 MMX나 3D-NOW 등 멀티미디어 처리 명령어 지원이 없고, 별도의 부동소수점 처리 명령어가 없는 ARM 또는 MIPS 등 휴대 정보 시스템에 사용되는 내장형 마이크로프로세서에서는 3D 그래픽 처리 과정의 약 50%를 차지하는 부동소수점 Geometry 단계의 처리가 매우 부담이 된다. 이는 type-1과 마찬가지로 정교한 3D 그래픽 컨텐츠 구현에 필요한 50만 polygon/sec 이상의 성능에는 턱 없이 부족한 성능이다. 즉 휴대 정보 시스템에서 더욱 향상된 3D 그래픽 가속 능력을 위해서는 Geometry 단계를 하드웨어로 가속하는 type-3과 같은 구조가 필수적이다.

본 논문에서는 휴대 정보기기 시스템에서 더욱 향상된 실시간 3D 그래픽 가속 능력을 갖는 SoC(System on a Chip) 구현을 위해 효과적인 Geometry 처리 IP(Intellectual Property) 구조를 연구하였다. 이를 기반으로 3D 그래픽 Geometry 처리 과정에 필요한 부동소수점 연산기를 설계하였으며, 휴대 3D 그래픽 국제 표준인 OpenGL-ES를 지원하는 부동소수점 Geometry 프로세서를 설계하였다[2].

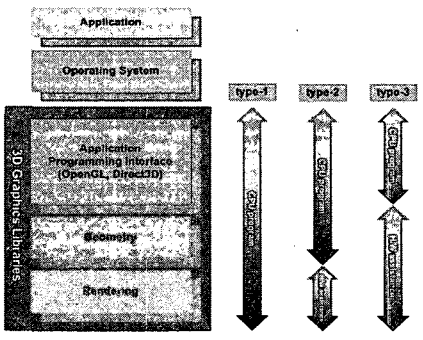

그림 1. 3D 그래픽 라이브러리 계층
Fig. 1. A class of 3D graphic library.

II. 3D 그래픽 Geometry 처리 과정

Geometry 처리는 호스트에 의해 처리된 모델 데이터의 정보들의 입력받음으로서 시작된다. 모델 데이터는 한 좌표의 정보, 각 좌표의 수직 방향을 나타내는 노말 벡터(Normal Vector), 물질과 광원의 Ambient, Diffuse, Specular 색 속성 정보와 각 단계에서 요구되는 파라미터들이 포함된다. Geometry 처리는 그림 2와 같이 모델 데이터에 대한 공간상의 위치 변환을 처리하는 Transformation 과정과 모델 데이터를 구성하는 각 정점의 빛에 대한 색 계산을 하는 Lighting 과정으로 구성되며, 일반적으로 Geometry 처리를 T&L 처리라고도 한다[3].

1. Transformation 처리 과정

Transformation 처리 과정은 모델의 좌표를 시점과 이동 정보로 변환하는 Model/View Transformation 과정, 3차원 View Volume으로 투영하는 Projection 과정, View Volume 밖에 위치하는 모델의 정점을 절단하는 Clipping 과정, 모델의 정점 좌표 x, y, z를 w로 나누는 Divide by W 과정과 2D 화면 좌표로 모델 좌표를 변환하는 ViewPort Transformation 과정으로 구성된다.

가. Model/View Transformation 과정

Model/View Transformation은 모델이나 그 모델 속의 특정 물체를 이동, 크기, 회전 변환으로 조작하는 과정인 Model Transformation과 모델을 바라보는 관측자의 관측점에 따라 모델을 회전 변환하는 과정인 View Transformation의 통합 과정이다. Model/View transformation 과정은 이동, 회전, 크기 조정 행렬을 곱셈 연

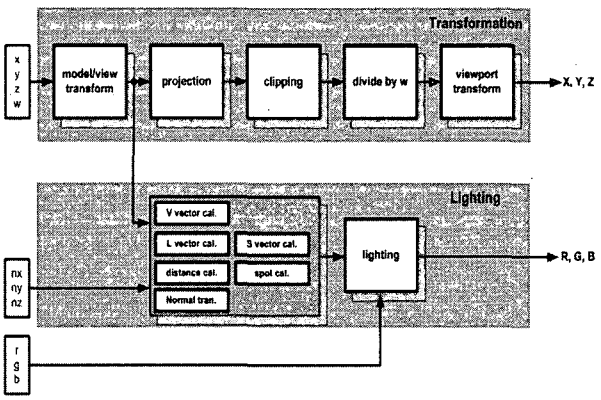

그림 2. 3D 그래픽 Geometry 처리 과정
Fig. 2. 3D graphic Geometry process.

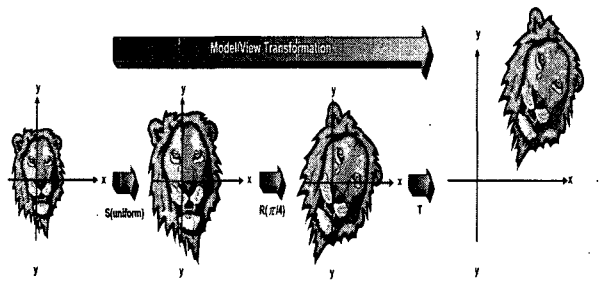

그림 3. Model/View transformation의 예
Fig. 3. A example of Model/View transformation.

산으로 결합한 4 * 4 Model/View 행렬 M과 x, y, z, w로 구성된 모델의 좌표 4 * 1 행렬을 곱하여 이동, 회전, 크기 조정 변환된 새로운 모델 좌표 4 * 1 행렬을 구하는 것이며, 그림 3은 Model/View transformation의 예를 나타낸다.

나. 투영 변환(Projection) 과정

투영 변환 단계에서는 3D 공간상에서 실제로 모델이 투영되는 6면체 영역인 View Volume을 정의하며, Model/View transformation 과정에 의해 변환된 모델이 View Volume의 클립좌표계(Clip Coordinate System)로 변환된다. 투영 변환은 직교 투영(Orthographic Projection)과 투시 투영(Perspective Projection)이 있다. 직교 투영은 모델의 좌표가 지정된 비율대로 정확히 스크린 상에 그려진다. 투시 투영은 원근법을 사용하여 멀리 떨어져 있는 모델은 같은 크기의 가까이 있는 물체보다 작게 보이는 효과를 낸다.

투영 변환은 직교 투영 행렬 또는 투시 투영 4 * 4 행렬과 모델의 좌표 4 * 1 행렬을 곱하여 새로운 모델 좌표 4 * 1 행렬을 구하는 것이다. 원근 투영 행렬의 마지막 행은 특정 수치를 가지는 벡터나 점을 포함하며, 투영 변환 후 새로운 모델 좌표의 w는 1이 아닐 수도 있으므로 이를 동차화하기 위해 x, y, z, w 값들을 w로 나눠주는 작업이 필요하다. 직교 투영의 경우 Model/View transformation 변환과 만찬가지로 투영 변환 후에도 w 성분에 영향을 주지 않고, w가 1인 동차 좌표를 유지한다.

다. Clipping 과정

투영 변환 후 모델을 구성하는 정점들의 좌표가 view volume 안쪽에 있는지 여부를 확인한다. view volume은 3D 공간에서 관찰자로부터 보이는 영역을 말하며, view volume 바깥에 있는 정점들은 관찰자로부터 보이지 않는 정점이므로 그림 4와 같이 절단(Clipping)해야 한다.

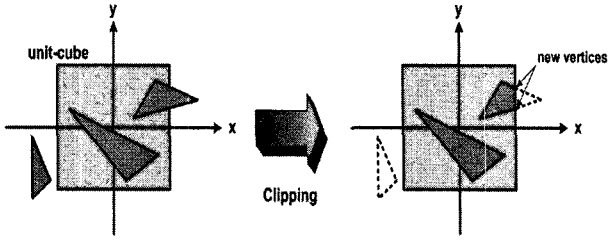

그림 4. View volume clipping 과정
Fig. 4. A view volume clipping process.

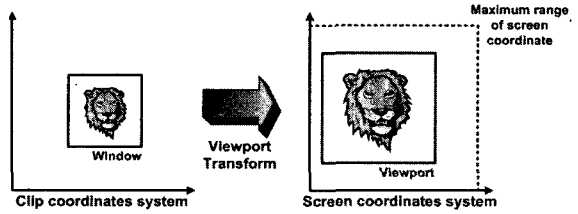

그림 5. Viewport 변환 과정
Fig. 5. A viewport transformation process.

라. Viewport 변환 과정

일반적으로 투영 변환에서 설정한 view volume의 폭과 높이가 실제적으로 이미지를 표현하는 스크린과 일치하지 않는다. 따라서 투영 변화에 의해 클립 좌표계로 변환된 모델의 좌표를 실제적인 스크린의 2D 윈도우 좌표계(Window Coordinate System)로 변환하는 과정이 필요하다. 이 변환 과정은 그림 5와 같이 view volume 내의 모델들이 윈도우 스크린 영역 안에 설정된 viewport로 좌표 변환되는 것을 의미한다.

2. Lighting 처리 과정

Lighting 처리 과정은 모델을 구성하는 정점의 수직 방향을 나타내는 노말 벡터의 변환, 광원에 대한 반사율을 계산하기 위한 방향 벡터인 L, S 벡터 계산, 광원의 감쇠 효과 계산을 위한 광원과 모델의 정점 간의 거리 D 계산, 집중 조명 광원 처리를 위한 Spot. effect 계산과 이를 이용하여 모델의 정점의 색 R, G, B를 계산하는 과정으로 구성된다.

3D 그래픽에서 모델의 정점의 색을 계산하는 조명 처리식인 수식 (1)은 광원이 어떻게 물체의 재질의 매개변수들과 상호작용을 하는지를 결정하고, 결과적으로 화면상에서 특정 물체가 점유하고 있는 정점들의 색상을 결정한다[4].

$$
\begin{aligned}
Vertex\ Color = & emission_{material} + \\
& ambient_{light\ model} * ambient_{material} + \\
& \sum_{i=0}^{n-1}\left(\frac{1}{k_c + k_l d + k_q d^2}\right)_i * (spotlight\ effect)_i * \\
& [ambient_{light} * ambient_{material} + \\
& (\max\{L \bullet n, 0\}) * diffuse_{light} * diffuse_{material} + \\
& (\max\{S \bullet n, 0\})^{shininess} * specular_{light} * specular_{material}]_i
\end{aligned} \quad (1)
$$

이것은 조명 효과가 광원으로부터 나온 빛에만 영향을 받고, 다른 표면에서 나온 빛에는 영향을 받지 않는다는 것을 의미한다. 조명 효과는 주변광, 난반사 그리고 정반사 성분에 의해 결정된다.

III. 부동소수점 연산기 연구 및 설계

1. 3D 그래픽 Geometry 처리 과정에서 요구되는 연산 연구

3D 그래픽 Geometry 처리 과정은 IEEE 754 단정도 부동소수점을 기반으로 하며, 처리 과정에 필요한 연산 종류는 사칙 연산 외에 더 복잡한 연산들을 요구한다. 표 1은 Geometry 각 단계에서 요구되는 연산의 종류를 나타낸다. 곱셈과 덧셈 연산은 Geometry 모든 단계에서 요구되며, 비교 연산은 View Volume 안쪽에 있는지의 여부를 검사하는 과정인 Clipping, 나눗셈 연산은 원근 투영 변환 후 모델의 좌표 동차화 과정과 조명 효과 계산에서 요구된다. 제곱근 연산은 방향 벡터들의 길이를 1로 만들어주는 벡터 정규화에 요구되며, 멱승 연산은 집중 조명광 효과 계산과 조명의 정반사 성분 계산에 요구된다.

다음의 항목은 3D 그래픽 Geometry 과정의 가속화를 위해 적합한 연산기의 특징과 이에 대하여 본 논문에서 부동소수점 연산기 설계 시 채택한 구조를 요약하였다.

표 1. Geometry 각 단계에서 요구되는 연산
Table 1. Geometry 각 단계에서 요구되는 연산.

요구 연산	Geometry 처리 과정
곱셈, 덧셈	Transformation, Lighting 전 과정
비교	Transformation의 Clipping 과정
나눗셈	Transformation의 동차화 과정 Lighting 전 과정
제곱근	Lighting의 벡터 정규화 과정
멱승	Lighting의 집중 조명 처리 과정 Lighting의 정반사 성분 처리 과정

1. 3D 그래픽 데이터의 내재된 병렬성을 이용할 수 있도록 다수의 연산기 구성에 용이해야한다.
2. 연속되는 연산 중 서로간의 데이터 의존성에 의한 성능 저하를 최소화해야한다.
3. 사용 빈도가 낮고, 지연 시간이 긴 나눗셈 연산과 제곱근 연산을 빠르게 처리 할 수 있어야 한다.

2. 부동소수점 연산기 설계

가. 부동소수점 덧셈(덧셈, 비교)/곱셈 연산기 설계

본 논문에서 제안하는 부동소수점 덧셈/곱셈기는 면적의 최소화를 위해, 지수 처리부와 라운딩, 정규화 부분 기능 유닛을 공유하여 설계하였으며, 데이터 의존성으로 인한 성능 저하를 막기 위해 파이프라인 단계를 일반적인 부동 소수점 덧셈기와 곱셈기의 파이프라인 단계인 3단계에서 2단계로 줄였다. 제안하는 부동소수점 덧셈/곱셈기는 일반적인 구조와 달리 라운딩과 정규화 과정의 배타성을 이용하여 두 과정을 병렬로 수행함으로서 이를 가수의 가감산 과정과 같은 파이프라인 단계에서 처리한다[5].

제안하는 부동소수점 덧셈/곱셈기는 그림 6과 같이 파이프라인 1 단계는 덧셈 연산의 경우 두 오퍼랜드의 지수 차를 계산하고, 곱셈 연산의 경우 두 오퍼랜드의 지수의 합을 계산하는 지수 처리 가감산기, 덧셈 연산에서 요구되는 지수의 차를 참조하여 가수를 정렬하는 베럴 쉬프터(Barrel Shifter), 두 오퍼랜드의 가수를 비교

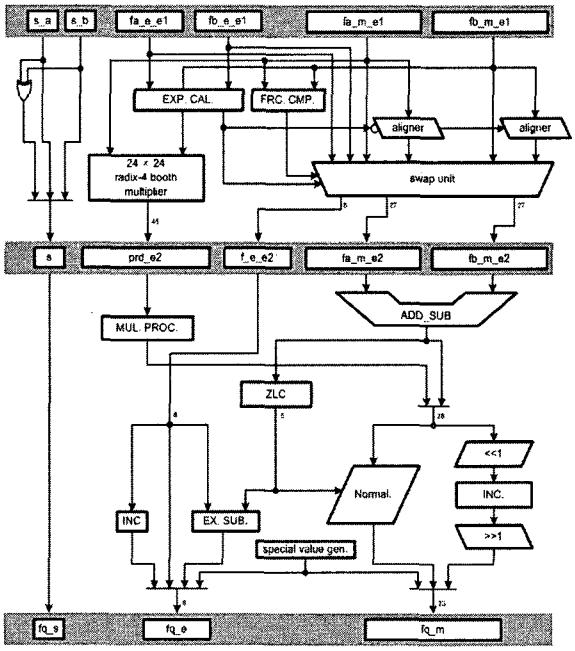

그림 6. 부동소수점 덧셈/곱셈기 구조
Fig. 6. A organization of floating point adder/multiplier.

하여 항상 큰 오퍼랜드가 왼쪽으로 향하게 하는 비교기와 위치 교환기(Swap Unit)와 라운딩을 위해 스티키(Sticky) 비트를 생성하는 스티키 비트 생성기와 곱셈 연산을 처리하기 위해 가수를 곱셈하는 24*24 Radix-4 booth 곱셈기로 구성되어있다. 파이프라인 2 단계는 덧셈 연산의 경우 두 오퍼랜드의 가수를 덧셈 하는 25-bit 고속 덧셈기와 곱셈 연산과 덧셈 연산이 공유하는 유닛으로서 정규화를 위해 곱셈, 덧셈기 결과의 선행 0의 개수를 검출하는 Leading Zero Counter, 라운딩을 수행하기 위한 증가기, 정규화를 수행하기 위한 베럴 쉬프터와 최종 지수를 계산하는 가감산기로 구성된다.

나. 부동소수점 나눗셈(역수) 연산기 설계

고속의 부동소수점 나눗셈(역수) 연산기를 구현하는 방안은 크게 두 가지 부류로 나뉜다. 첫째로 덧셈/뺄셈과 쉬프트 동작을 조합하여 구현하는 방식으로 복원(restoring) 알고리즘, 비복원(nonrestoring) 알고리즘, SRT 알고리즘이 있다. 둘째는 승산기를 이용하여 나눗셈을 수행하는 방식으로 분모, 분자에 동일한 수를 곱하는 수렴(convergence) 방식과 분모의 역수를 구해 곱하는 NR (Newton-Raphson) 방식이 있다. 본 논문에서는 3D 그래픽 처리 특성상 나머지 값이 필요하지 않고, 적은 수행 단계의 파이프라인 수행이 가능하며, 역수 계산에 유리한 NR 방식으로 나눗셈(역수) 연산기를 설계하였다.

본 논문에서 나눗셈(역수) 연산기 설계 시 사용한 NR 알고리즘은 테일러 급수를 이용한 역수 연산 근사식인 수식 (2)을 사용하였다[6].

$$\frac{1}{Y} \simeq (2-AY)A,\ A=\frac{(Y_h-Y_l)}{Y_h^2} \qquad (2)$$

그림 7과 같이 역수 알고리즘 수식 (2)을 하드웨어로 구현하기 위해서 4 단계 과정으로 구성할 수 있다.

첫 단계에서 $1/Y_h^2$은 주소 값이 8비트인 룩업 테이블에 의해 구해지며, 룩업 테이블은 엔트리의 수가 128개이고. 13비트의 데이터로 구성되어 총 208Byte의 크기를 갖는다. 두 번째 단계에서 A는 Y_h-Y_l과 $1/Y_h^2$을 곱하여 구해진다. 곱셈기 M1의 크기는 24 * 13이 되며 결과 값인 A는 MSB를 제외한 상위 15비트로 구성된다. 세 번째 단계에서 AY가 계산되며, 곱셈기 M2의 크기는 28 * 15가 되며 결과 값인 AY는

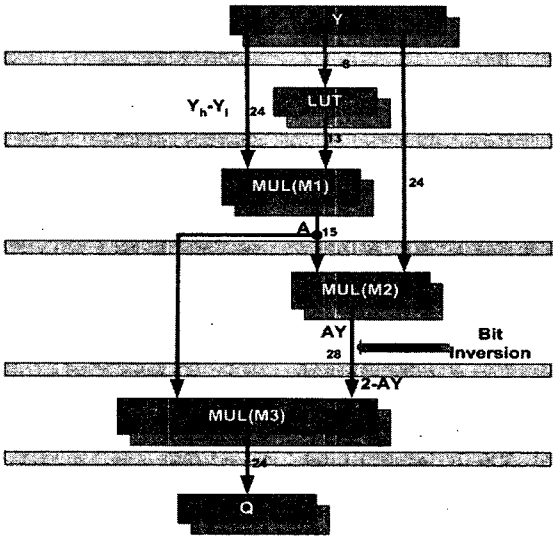

그림 7. 부동소수점 나눗셈(역수) 연산기 구조
Fig. 7. A organization of floating point division(reciprocal) unit.

MSB를 제외한 상위 28비트로 구성된다. 여기서 $2-AY$는 AY의 bit inversion에 의해서 구한다. 마지막으로 최종 몫 $A(2-AY)$는 A와 $2-AY$의 곱셈으로 구해진다. 곱셈기 M4의 크기는 15 * 28이 되며 결과 값인 Q는 MSB를 제외한 상위 24비트로 구성된다.

다. 부동소수점 역 제곱근 연산기 설계

부동소수점 역 제곱근 연산을 하드웨어로 구현하기 위한 여러 알고리즘이 있지만 일반적으로 긴 입력 레이턴시나, 큰 용량의 메모리의 사용을 요구된다. 설계된 부동소수점 역 제곱근 연산기는 변형된 피 연산자와 승산을 이용한 룩업 테이블을 생성해 초기 근사화를 사용하였다. 이 초기 근사화 과정 후에 변형된 NR 반복법은 더 정확한 역 제곱근을 생성하게 된다. 이 초기 근사화와 변형된 NR 반복법은 하드웨어의 딜레이, 면적 그리고 전력 소모를 줄일 수 있다[7]. 그림 8은 설계된 부동소수점 역 제곱근기 연산기이다.

라. 부동소수점 멱승 연산기 설계

멱승 연산은 음영 처리과정 중 정반사 성분 $(n \cdot h)^{m_s}$와 집중 조명광 $(-l \cdot s_{dir})^{2^{exp}}$을 연산하기 위해 사용된다. 일반적으로 멱승 연산을 구하는 방법은 여러 가지이나 대부분의 경우 곱셈, 나눗셈 연산이 필요하여, 속도면에서 크게 떨어진다. 그러므로 본 설계에서는 그림 9와 같이 음영 처리 프로세서 구조에 적합하고, 속도 면에서 가장 좋은 룸 테이블 방식을 사용했다[8].

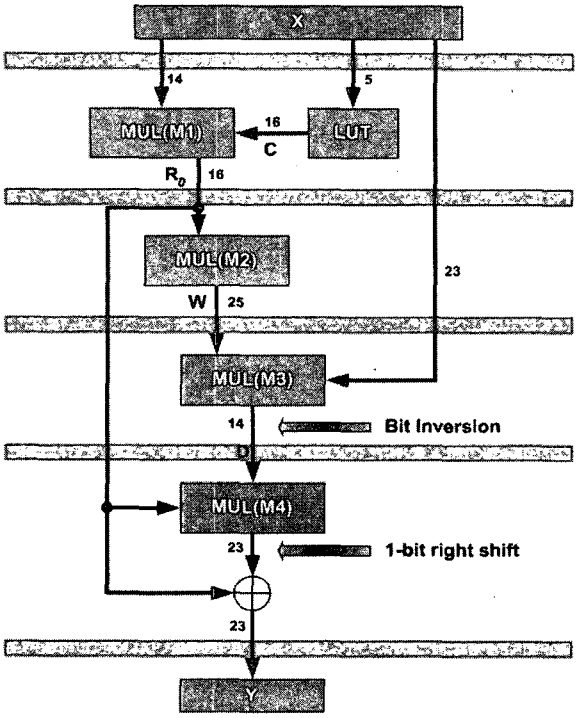

그림 8. 부동소수점 역 제곱근 연산기 구조
Fig. 8. A organization of floating point inverse square root unit.

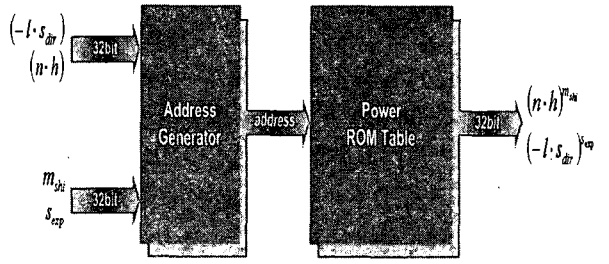

그림 9. 부동소수점 멱승 연산기 구조
Fig. 9. A organization of floating point power unit.

롬 테이블 방식의 멱승 연산기는 $(n \cdot h)$값 또는 $(-l \cdot s_{dir})$의 값과 지수인 m_{sh} 또는 $s_{\exp}$ 값을 입력으로 받아 양자화 된 멱승 연산 결과 값이 저장되어 있는 롬 주소를 발생시켜 해당하는 $n(\cdot h)^{m_s}$ 또는 $(-l \cdot s_{dir})^{s_{\exp}}$을 얻는 방식이다.

IV. 3D 그래픽 Geometry 프로세서 구조 연구 및 설계

1. 3D 그래픽 Geometry 프로세서 구조 연구

Geometry 처리 과정은 각 처리 단계마다 모델의 좌표 데이터와 처리 방법의 병렬성이 크게 내재되어있다. 그러므로 이를 최대한으로 이용하여 동시에 여러 데

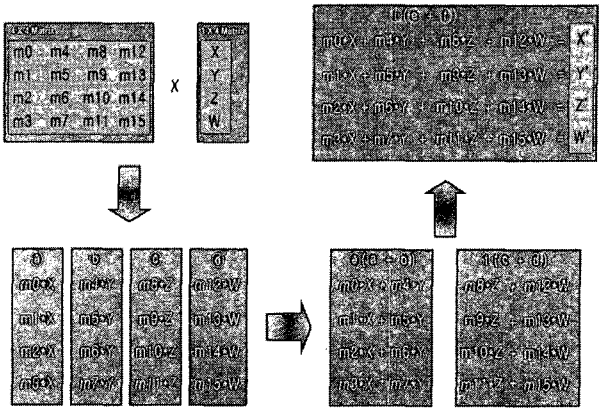

그림 10. 128-bit SIMD 구조를 적용한 행렬 곱셈 연산
Fig. 10. A matrix multiplication in 128-bit SIMD architecture.

이터를 파이프라인으로 빠르게 처리하는 병렬구조가 Geometry 프로세서에 적합하다. 본 논문에서는 데이터 패스의 크기가 비교적 작고, 제어부가 간단하면서도 3D 그래픽 데이터의 병렬성을 최대한 이용할 수 있는 SIMD 구조를 채택하였다. 그림 10은 Geometry 처리 과정에서 주가 되는 4 * 4 행렬과 1 * 4 행렬 곱셈 연산을 128-bit SIMD 구조로 처리하는 과정을 나타낸다.

4*4 행렬과 1*4 행렬 곱셈 연산을 128-bit SIMD 구조를 적용한 프로세서에서 처리할 경우, 그림 10과 같이 ⓐ ~ ⓘ의 과정으로 처리할 수 있으며, 이는 7개의 명령어 수행으로 4 * 4 행렬과 1 * 4 행렬 곱셈 연산 처리할 수 있음을 의미한다.

프로세서의 파이프라인 구조 역시 병렬 구조 채택과 마찬가지로 Geometry 프로세서에 성능을 크게 영향을 미친다. Geometry Transformation 프로세서의 명령어 파이프라인이 데이터 의존성(Data Dependency)이나, 처리 유닛 의존성(Resource Dependency) 등 파이프라인 해저드(Pipeline Hazard)로 정지(Stall) 될 경우 연속적인 반복 수행(Loop)으로 구성된 Geometry 처리 과정은 상당히 큰 성능 저하를 야기하게 된다.

Geometry 프로세서의 파이프라인은 명령어 해석 단계인 ID, 부동소수점 연산 단계인 EX1, EX2, EX3 와 연산 결과를 저장하는 WB 단계로 구성된다. 일반적인 부동소수점 연산은 3개의 수행 단계를 갖는다. 이로 인하여, ⓔ ~ ⓘ 과정에서 데이터 의존성을 해결하기위한 데이터 전방전달(Data Forwarding)이 파이프라인 정지 없이 수행될 수 없다. 이러한 데이터 의존성에 의한 파이프라인 정지는 부동소수점 연산기의 수행 단계가 길어질수록 많이 발생하며, 그림 10의 행렬 곱셈 연산뿐만 아니라 데이터 로드, 스토어(data load/store) 등 Geometry 처리 전 과정에서 수차례 발생한다.

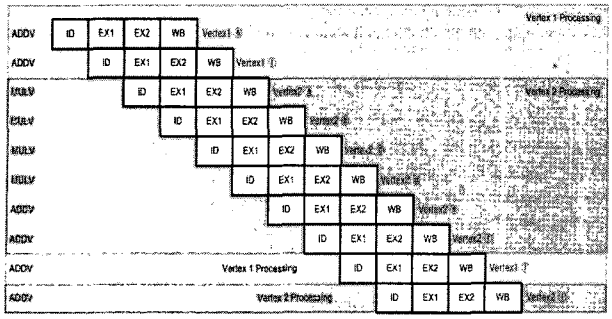

그림 11. 제안하는 geometry 프로세서 파이프라인 구조
Fig. 11. Proposed pipeline architecture of the geometry processor.

본 논문에서는 파이프라인 해저드로 인한 Geometry 프로세서의 성능 저하를 줄이기 위해서 그림 11과 같은 파이프라인 구조를 제안하였다.

본 논문에서 제안하는 그림 11의 Geometry 프로세서 파이프라인 구조는 3장에서 설계된 2단계 부동소수점 연산기를 기반으로 4단계로 설계하였으며, 모델의 정점 두 개를 중첩하여 처리하는 구조로 데이터 의존성으로 인한 파이프라인 정지를 제거하였다. 본 논문에서 제안한 파이프라인 구조는 Geometry Transformation 및 Lighting 처리 전 과정에서 두 개의 정점을 중첩하여 처리함으로서 단 한 번의 파이프라인 정지도 발생하지 않는다.

2. 3D 그래픽 Geometry 프로세서 설계

본 논문이 제안하는 Geometry 프로세서는 앞에서 언급한 연구를 기반으로 설계되었다. 즉 Geometry 처리 과정의 병렬성을 최대한 이용하여 처리할 수 있도록 128-bit, 96-bit, 64-bit SIMD 형식의 명령어를 정의하였으며, 이를 지원하기 위하여 3장에서 설계된 부동소수점 연산기들을 SIMD 형식으로 구성하였다. 또 파이프라인 구조와 레지스터 파일 구조를 SIMD 명령어의 효율적 수행을 위해 최적화 설계하였다.

본 논문에서 정의한 명령어는 MULV, ADDV, CMPV, CONV, SQRT, POW, LDRV, STRV, MOVV 명령어로 구성되며, 명령어의 끝의 'V는 128-bit, 96-bit, 64-bit SIMD 연산의 지원을 의미한다.

SIMD 형식 명령어는 기본적으로 4 단계 파이프라인 수행 단계를 가진다. 단 역수 연산 명령어 CONV와 역제곱근 연산 명령어 SQRT의 수행은 6 단계의 파이프라인 단계를 가진다. 그림 12는 Geometry 프로세서의 SIMD 명령어 수행 파이프라인 단계를 나타낸다.

MULV, ADDV 등과 같은 부동소수점 연산 명령어들은 부동소수점 연산의 긴 지연 시간으로 인해 다수의

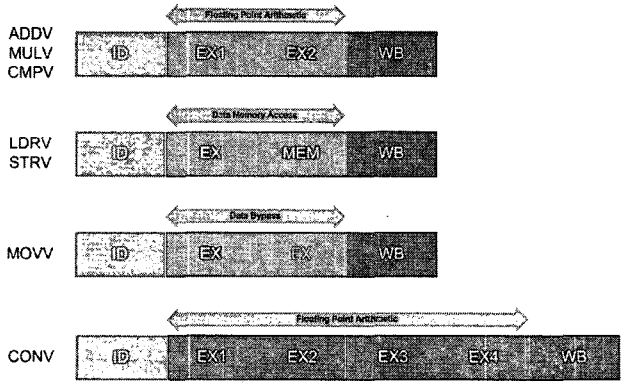

그림 12. 제안하는 SIMD 명령어의 파이프라인 단계
Fig. 12. A pipeline stage of the proposed SIMD instructions.

EX(Execution Stage) 단계를 갖으며, 특히 부동 소수점 역수 연산은 네 개의 EX 단계로 파이프라인이 구성된다. LDRV, STRV와 같은 메모리 전송 명령어는 메모리의 주소와 제어신호를 발생하는 EX 단계와 메모리와 데이터를 주고받는 MEM(Memory Stage)단계로 파이프라인이 구성되며, MOVV 명령어는 프로세서의 제어 및 파이프라인 구성의 편의를 위하여 두 번의 Bypass 단계를 갖는 4 단계 파이프라인으로 구성된다.

제안하는 Geometry 프로세서는 앞 장에서의 파이프라인 연구를 기반으로 데이터 의존성에 의해 생기는 파이프라인 멈춤을 제거하기 위해 두 개의 정점 데이터를 중첩하여 파이프라인 수행을 한다. 두 개의 정점 데이터를 중첩하여 파이프라인 수행을 위해서는 프로세서의 레지스터 파일이 이에 맞는 적합한 구조로 구성되어야 한다. Geometry 프로세서는 총 73개의 32-bit 레지스터로 구성되며, 이를 그룹화 하여 효율적으로 접근할 수 있는 레지스터 파일 구조를 가진다. 이는 정점 데이터를 하나씩 처리 할 때보다 더 많은 레지스터의 개수가 필요하게 되지만, 파이프라인이 멈추지 않고 처리할 수 있도록 지원하여 결과적으로 Geometry 과정의 처리 속도 향상을 가져온다. 그림 13은 Geometry 프로세서의 레지스터 파일 구조를 나타내며, 표 2에서는 레지스터 파일 그룹에 저장되는 데이터를 정리하였다.

Geometry 프로세서는 그림 14와 같이 SIMD 데이터 패스구조를 갖는다. Geometry Transformation 프로세서는 크게 명령어 디코더, 전방 전발 회로, 제어 회로, 레지스터 파일, 부동 소수점 연산 회로, 로드/스토어 회로로 구성된다. 명령어 디코더는 호스트 프로세서로부터 전달받은 명령어를 해석하는 회로이다. 전방 전달 회로는 데이터 의존성이 발생하였을 때 데이터를 전방 전달하기 위한 회로이며, 제어 회로는 Geometry 프로세서

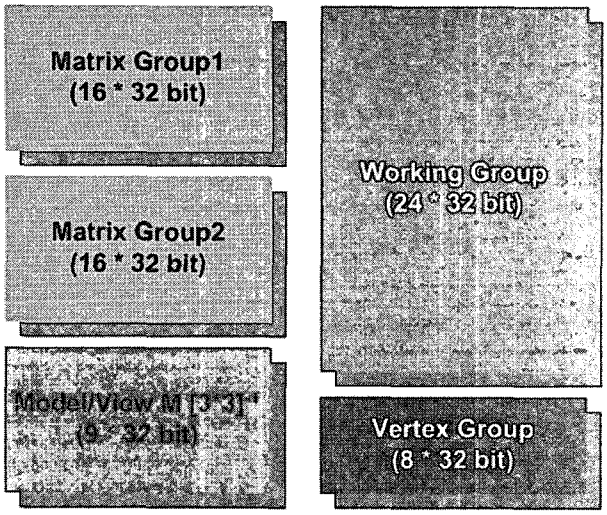

그림 13. 제안하는 레지스터 파일 구조
Fig. 13. The proposed register file architecture.

표 2. 레지스터 그룹의 저장 데이터
table 2. 3D graphics data of each register group.

레지스터 그룹	저장 데이터
Matrix Group1 (MG1)	Model/View Matrix 4*4
Matrix Group2 (MG2)	Projection Matrix 4*4
Model/View $M[3*3]^{-T}$(NMG)	Normal Vector Matrix 3*3
Working Group (WG)	Temporary Data Color Data
Vertex Group (VG)	Vertex Data

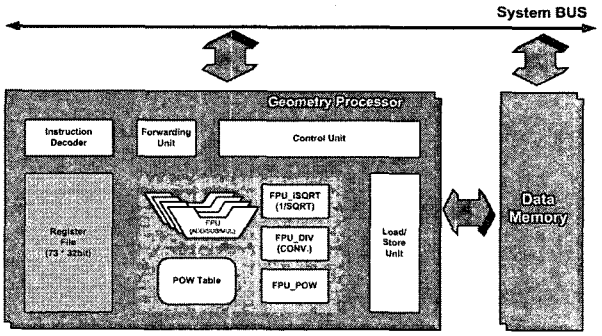

그림 14. 제안하는 3D 그래픽 Geometry 프로세서
Fig. 14. The proposed 3D graphics geometry processor.

의 파이프라인과 데이터패스를 제어하는 역할을 한다. 레지스터 파일은 32*73 비트의 크기를 가지며, 여러 4*4 변환 행렬, 두 정점의 좌표와 색 정보, 임시 데이터를 저장하며, 부동소수점 연산 회로는 네 개의 부동소수점 덧셈/곱셈기, 한 개의 역수 연산기, 한 개의 역 제곱근 연산기와 한 개의 멱승기로 구성돼있다. 로드/스토어 유닛은 데이터 메모리로부터 데이터를 로드 또는 스토어 하는데 필요한 주소와 메모리 제어 신호를 생성하며, SIMD 데이터를 정렬하는 역할을 한다.

V. 검증 시스템 설계 및 성능평가

3D 그래픽 데이터는 3D 그래픽 가속기에서 Geometry 처리 및 Rasterization 처리를 거친 후 메모리에 저장되고 이를 화면으로 보내어 원하는 영상을 표현한다. 그래서 본 논문의 3D 그래픽 Geometry 프로세서를 실험하기 위해서는 Geometry 프로세서를 제어하는 호스트 프로세서, 폴리곤 데이터를 받아 각 픽셀의 색상 정보를 모아 한 화면의 영상으로 구성하는 Rasterization 처리기와 이를 출력해주는 디스플레이 장치가 필요하다. 그림 15는 3D 그래픽 Geometry 프로세서의 실험 환경으로 호스트 프로세서는 VHDL을 이용한 동작 모델링을 통해 PCI 버스 타입의 FPGA에 구현된 Geometry 프로세서에게 명령어와 3D 그래픽 데이터를 전달한다. Rasterization 처리기는 Mesa 3D 라이브러리를 기반으로 3D 그래픽 Rasterization 파이프라인을 프로그램으로

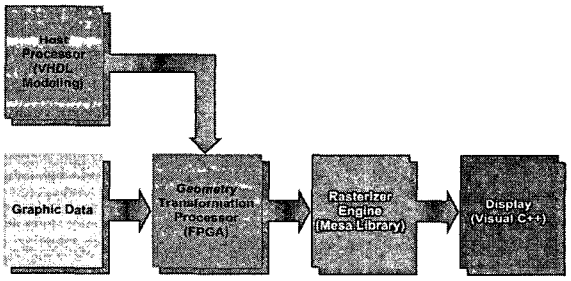

그림 15. 3D 그래픽 Geometry 프로세서의 실험 환경
Fig. 15. Experiment environment of the 3D graphics geometry processor.

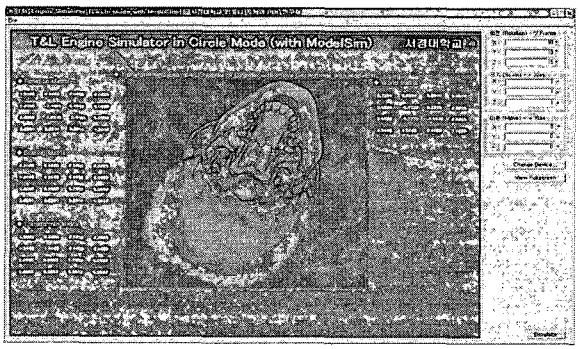

그림 16. 3D 그래픽 Geometry 프로세서의 실험 환경 GUI
Fig. 16. Experiment environment GUI of the 3D graphics geometry processor.

구현하였다[9]. 디스플레이 부분은 Visual C++의 그래픽 라이브러리를 이용하여 구현하였다.

그림 16은 실험 환경의 GUI로 3D 그래픽 실험 데이터인 SKULL 메쉬의 Geometry 처리 과정을 수행한다. 실험 데이터인 SKULL 메쉬는 60339개의 폴리곤으로 구성되어있다.

3D 실험 데이터인 SKULL 메쉬의 한 정점에 대하여 Transformation 과정에서 각 단계를 수행하는데 걸리는 사이클의 수를 표 3에 정리하였다. 이동, 회전 크기 변환 4*4 행렬을 곱하여 4*4 Model/View 행렬을 생성하는 과정은 다른 여러 단계보다 비교적 긴 수행시간을 갖으나, 본 논문에 제안하는 Geometry 프로세서는 4 * 4 Model/View 행렬을 한 장면마다 한 번 생성하여 레지스터 파일에 저장함으로서 매 정점마다 이를 생성하지 않는다.

Geometry 프로세서의 FPGA(Xilinx-Vertex2) 합성 결과는 160K gate의 면적과 80 MHz의 동작 속도를 확인하였으며, 80 MHz 동작 시 초당 4M 개의 폴리곤을 Transformation 처리할 수 있으며. 광원의 개수와 종류에 따라서 초당 1M ~ 1.6M 개의 폴리곤을 T&L 처리할 수 있다.

표 4는 타 Geometry 프로세서와 본 논문이 제안한 Geometry 프로세서의 성능 비교를 정리하였다. 성능 비교 관점은 프로세서 동작 주파수와 초당 폴리곤 처리 개수이며, 비교 대상은 VFP(Vector Floating-point Processor)를 내장한 ARM10 마이크로프로세서, Satine (Kaist), Z3D(Mitsubishi) 프로세서와 SH4 (Hitachi) 프로세서이다.

표 3. Geometry 프로세서의 변환 처리 수행 사이클 수
table 3. The number of execution cycle of transformation process.

Transformation 과정	부동 소수점 연산	수행 사이클 수
Model/View 변환	곱셈 4 회 덧셈 3 회	7 cycles
투영 변환	곱셈 4 회 덧셈 3 회	7 cycles
동차화 과정(1/w)	역수 연산 1회 곱셈 1회	4 cycles
Clipping	비교 연산 1회	1 cycle
Viewport 변환	곱셈 1 회 덧셈 1 회	2 cycles

표 4. 타 프로세서와의 성능 비교
table 4. Performance comparison result with others processor.

프로세서	Geometry 처리 성능
ARM10 + VFP[10]	100K polygons/sec @150MHz
SATINE[11]	400K polygons/sec @200MHz
Z3D[12]	250K polygons/sec @30MHz
SH4[13]	500K polygons/sec @200MHz
제안하는 eometry 프로세서	1M ~ 1.6M polygons/sec @200MHz

표 5. 소모전력 및 게이트수 측정결과(Hynix 0.25um)
Table 5.

Type		Original		After Clock Gating	
		Cell Count	Power [mW]	Cell Count	Power [mW]
Cell Internal	Combinational	39,420	32.9	32,842	33.0
	Sequential	4,065	35.4	4,258	15.1
	Other	3,424	0.9	3,424	0.9
	Sub Total	-	69.2	-	49.0
Net Switching		-	14.4	-	14.7
Total Dynamic		-	83.6	-	63.8
			100%		76.23%
Cell Leakage		-	6.8811 [uW]	-	6.9545 [uW]

표 5는 Hynix 0.25um CMOS공정과 라이브러리를 이용한 소모전력 평가와 게이트수를 나타낸다. 소모전력 측정은 50MHz에서 수행되었다. 측정결과 83.6mW의 전력소모를 보였다.

Ⅳ. 결 론

제안하는 3D 그래픽 Geometry 프로세서는 Geometry 과정에 최적화 되도록 3D 모델을 구성하는 폴리곤의 정점 좌표와 색상을 벡터 단위로 처리할 수 있도록 128-bit, 96-bit, 64-bit SIMD 구조를 갖는다. 또한 프로세서의 파이프라인은 데이터 의존성에 의해 파이프라인 멈춤을 방지하기 위하여 정점 데이터를 두 개씩 중첩하여 처리하도록 최적화하였다.

설계된 Geometry 프로세서는 Xilinx-Vertex2 FPGA에서 160K gate의 면적으로 구현되었으며, 80 MHz의

동작주파수 환경에서 실제 3D 그래픽 데이터를 이용하여 Geometry 과정의 성능 측정 실험을 하였다. 실험 결과 80 MHz의 동작주파수에서 초당 1M ~ 4M개의 프리미티브 처리 성능이 확인되었으며, 이는 타 Geometry 가속 프로세서에 비하여 평균 200% ~ 300%의 Geometry 처리 가속 성능이다.

참 고 문 헌

[1] Makoto Awaga, Tatsushi Ohtsuka, Hideki Yoshizawa and Shigeru Sasaki, "3D Graphics Processor Chip Set," *IEEE Micro*, pp.37-41, dec., 1995.

[2] M. Woo, J. Neider, T. Davis, D. Shreiner *"OpenGL Programming Manual,"* Addison & Wesley, 1997.

[3] J.D Foley, A. Dam, S. K. Feiner, and J. F. Hughes, *"Computer Graphics Principles and Practice,"* Second Edition, Addison-Wesley, 1990.

[4] Tomas Akenine-Moller, Eric Haines *"REAL-TIME RENDERING,"* Second Edition, AK Peters Natick, Massachusetts, 2003.

[5] Woo Kyeong Jeong, "A SIMD-DSP/FPU for High-Performance Embedded Microprocessors," Phd Thesis, Yonsei University, Dec., 2002.

[6] J. C. Jeong, W. C. Park, W. Jeong, T. D. Han, M. K. Lee "A Cost-Effective Pipelined Divider with a Small Lookup Table", *IEEE Transactions on Computers*, pp.489-495. 2004.

[7] Michael J, Schulte, Kent E, "High-Speed Inverse Square Roots," 14th IEEE Symposium, pp.124-131, 1999.

[8] Hyun-Chul Shin, Jin-Aeon Lee, and Lee-Sup Kim, "A Hardware Cost Minimized Fast Phong Shader," *IEEE TRANSACTIONS ON VLSI SYSTEMS*, Volume 9, Issue 2, pp.297-304, April 2001.

[9] Mesa library, http://www.ssec.wisc.edu/~brianp/Mesa.html.

[10] Won-Suk Kim, "The Implementation of Geometry Accelerator Simulator for 3D Graphic Accelerator Hardware Design," Yonsei Univ., Master Thesis, 2003.

[11] Ju-Ho Sohn "Design and Optimization of Geometry Acceleration for Portable 3D Graphics," KAIST, Master's Thesis, 2003.

[12] Masatoshi Kameyama, Yoshiyuki Kato, Hitoshi Fujimoto, Hiroyasu Negishi, Yukio Kodama, Yoshitsugu Inoue, Hiroyuki Kawai, "3D graphics LSI core for mobile phone Z3D," Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, pp.60-67, 2003.

[13] F. Arakawa, O. Nishii, K. Uchiyama, and N. Nakayama, "SH4 RISC multimedia microprocessor," *IEEE Micro*, vol. 18, no. 2, pp.26-34, April 1998.

저 자 소 개

남 기 훈(학생회원)
1999년 2월 서경대학교 컴퓨터과학과 학사.
2001년 2월 서경대학교 컴퓨터과학과 석사.
2001년 ~ 현재 서경대학교 컴퓨터과학과 박사과정.
<주관심분야 : Diagnostic Function Test, 마이크로프로세서, 암호프로세서, Embedded System, 3D Graphics System>

하 진 석(학생회원)
2001년 2월 서경대학교 컴퓨터과학과 학사.
2003년 2월 서경대학교 컴퓨터공학과 석사.
2003년 ~ 현재 서경대학교 컴퓨터공학과 박사과정.
<주관심분야 : 저전력 마이크로프로세서, Computer Arithmetic, 암호프로세서, Embedded System, 3D Graphics System >

곽 재 창(정회원)
1983년 2월 연세대학교 문학사
1989년 12월 Univ. of Iowa 전산학 석사
1993년 8월 Univ. of Iowa 전산학 박사
1995년 ~현재 서경대학교 컴퓨터과학과 부교수.
<주관심분야 : Network Traffic Control, QoS, Realtime Scheduling, Embedded System>

이 광 엽(정회원)
1985년 8월 서강대학교 전자공학과 학사.
1987년 8월 연세대학교 전자공학과 석사.
1994년 2월 연세대학교 전자공학과 박사.
1989~1995년 현대전자 선임연구원
1995년~현재 서경대학교 컴퓨터공학과 부교수.
<주관심분야 : 마이크로프로세서, 암호프로세서, Embedded System, 3D Graphics System>