논문 2007-44SD-8-13

컬링과 클리핑을 포함한 3D 그래픽스 래스터라이져 설계

(A Design of a 3D Graphics Rasterizer with culling and clipping)

이 광 엽*, 구 용 서**

(Kwang-Yeob Lee and Yong-Seo Koo)

요 약

본 논문은 효율적인 3차원 그래픽스를 위해 컬링과 클리핑을 포함한 래스터라이져를 설계하였다. 제안하는 래스터라이져는 모바일 환경을 위해 구현하였고, 프러스텀 컬링, 백 페이스 컬링, Y축 클리핑, X축 클리핑을 처리한다. 래스터라이져는 트라이앵글 셋업, 에지 워크, 스팬 프로세서 유닛으로 구성된다. 컬링, 클리핑을 포함한 래스터라이져의 각 유닛으로 설계하였다. 래스터라이져는 16 비트 깊이 값과 16 비트 컬러 값을 갖는 고라우드 쉐이딩을 지원한다. 제안한 래스터라이져는 52M pixels/sec의 처리 능력을 갖는다.

Abstract

In this paper, we designed 3D graphics rasterizer with a culling and clipping for the efficient 3D graphics accelerator. The proposed rasterizer is implemented for the mobile system and process frustum culling, back face culling, Y-axis clipping and X-axis clipping. The rasterzier consists of triangle setup, edge walk and span process unit. Each unit of rasterzier is designed with a culling and clipping. It supports goraud shading with 16 bits depth values and 16 bits color values. The estimated performance of proposed rasterizer is 52M pixels per second.

Keywords: 3D Graphics, Rasterizer, Culling, Clipping

I. 서 론

1950년대의 그래픽스는 흑백 문자를 이용하여 라인 프린터, 텔렉스, 오실로스코프 등 출력장치에 그림을 그려냈다. 1960년대의 그래픽 출력은 벡터 그래픽 장비로 모든 물체를 선의 집합으로 표현하는 정도였다. 1970년대에 이르러 그래픽에 관련한 알고리즘이 개발되었고, 그래픽 국제 표준에 대한 필요성이 논의되었다. 1980년대에 개인용 컴퓨터와 그래픽 장비가 보급되었으며 실리콘 그래픽스 사에서 Geometry Engine을 출시하였다. 1990년대는 고해상도 컬러 그래픽을 지원하는 그래픽 카드가 개발됐다. 2000년대에 이르러 3D 그래픽스의 활용 분야는 더욱 넓어져 게임, 영화 등에서 더욱 복잡한 그래픽을 표현하고 있다. 또한 PDA, 핸드폰 등의 모바일용 장비에서 3D 그래픽스의 활용이 증가하고 있다.

3D 그래픽스는 많은 데이터 연산을 요구한다. 3D 그래픽스 처리 초기에는 CPU가 모든 연산 처리를 담당하였으나, 더욱 정교한 3D 그래픽스의 표현을 위해 데이터 연산 처리량이 증가였고, CPU만으로 처리하기에는 큰 부담이 되었다. 복잡도 높은 모델 데이터를 빠르게 처리하기 위해 별도의 그래픽스 가속 하드웨어 설계의 필요성이 요구되었다.[1~4]

그래픽 가속 하드웨어 처리 과정은 크게 지오메트리 과정과 래스터라이제이션 과정으로 구분된다. 모델 정점의 좌표 변환, 이동, 크기 변환과 광원을 반영하여 색상을 계산하는 것이 지오메트리 과정이고, 지오메트리 과정에서 변환된 모델의 데이터로 모델의 색상을 보간

* 평생회원, 서경대학교 컴퓨터공학과
(Dept. of Computer Engineering Seokyeong Univ.)
** 평생회원-교신저자, 서경대학교 전자공학과
(Dept. of Electronic Engineering Seokyeong Univ.)
※ 본 논문은 서울특별시 나노혁신 클러스터 사업의 지원과 한국전자통신연구원 SoC 산업진흥센터의 지원으로 제작되었습니다.
접수일자: 2007년6월7일, 수정완료일: 2007년8월14일

하는 것이 래스터라이제이션 과정이다.

Ⅱ. 본 론

1. 래스터라이제이션

래스터라이제이션은 주사 변환(Scan Conversion) 단계와 실감 영상 처리(Realization) 단계로 구분한다. 주사변환 단계는 트라이앵글 셋업, 에지 워크, 스팬 프로세스로 세분화한다. 실감 영상 처리 단계는 텍스쳐 매핑, 알파 블렌딩, Z-버퍼링 등으로 세분화한다.

주사 변환 단계는 세 개의 좌표와 색상의 값으로 삼각형을 형성하고, 보간 연산(Interpolation processing)으로 삼각형 내부의 각 픽셀의 정보들을 계산한다. 실감 영상 처리 단계는 모델에 추가적인 효과를 덧입혀 현실감을 높이는 과정이다. 텍스쳐 매핑으로 모델의 면을 현실감 높게 표현할 수 있고, 알파 블렌딩으로 모델의 투명도를 표현한다. Z-버퍼링은 모델을 높이 값을 비교하여, 모델이 중복되었을 때 겹침 현상을 방지한다.

[그림 1]은 래스터라이제이션을 2 단계로 구분하여 도시화한 것이다. 일반적으로 주사 변환 유닛의 트라이앵글 셋업, 에지 워크, 스팬 프로세스 과정을 처리하고, 실감 영상 처리 유닛의 텍스쳐 매핑, Z-버퍼링, 알파 블렌딩 과정을 처리한다. 래스터라이제이션 처리 단계는 처리 속도와 데이터 의존성에 따라 처리 순서의 변경이 가능하다.[1~4]

본 논문은 [그림 1]의 주사 변환 유닛을 설계하였다. 주사 변환 유닛의 기본 구조인 트라이앵글 셋업, 에지 워크, 스팬 프로세스 과정에 컬링과 클리핑 유닛을 포함하였다.

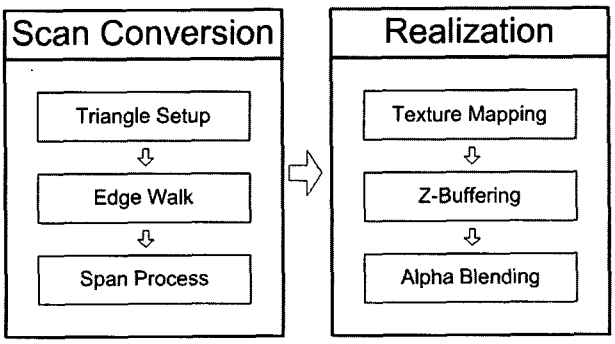


그림 1. 래스터라이제이션 파이프라인
Fig. 1. Rasterization Pipeline.

2. 제안하는 래스터라이져 구조

[그림 2]는 기존 방식의 컬링과 클리핑 방식으로 3D 그래픽 파이프라인에서는 지오메트리 후단부에 존재하며 일부 방식에서는 지오메트리 후단부와 래스터라이저 전단부에 나누어 존재한다. 본 논문에서는 래스터라이져의 파이프라인에서 컬링과 클리핑 처리 시간이 래스터라이져의 처리 시간에 종속되도록 [그림 3]과 같이 래스터라이져에서 컬링과 클리핑을 3단계로 나누어 처리하도록 하였다. 이 방법은 기존의 방식에서는 컬링과 클리핑에 별도의 처리 클럭사이클을 필요로하지만 제안하는 방법은 래스터라이져 처리 클럭사이클에 추가 없이 컬링 및 클리핑 실행이 가능하다.

[그림 4]는 제안하는 래스터라이져를 세부적으로 도시화한 것이다. 트라이앵글 셋업 과정은 삼각형의 각 정점의 위치를 정의한다. 이때 화면 영역 외의 삼각형

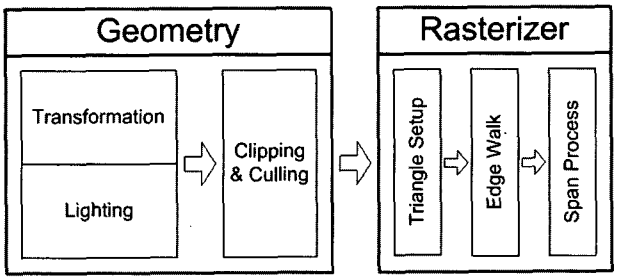


그림 2. 3D 그래픽스 파이프라인
Fig. 2. 3D Graphics Pipeline.

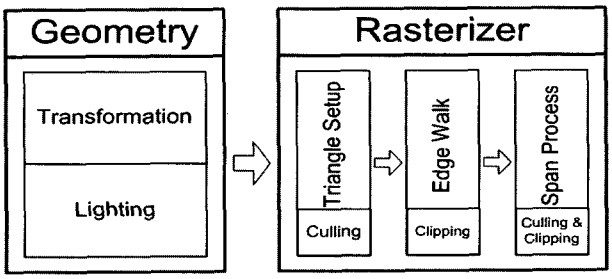


그림 3. 제안하는 3D 그래픽스 파이프라인
Fig. 3. Proposed 3D Graphics Pipeline.

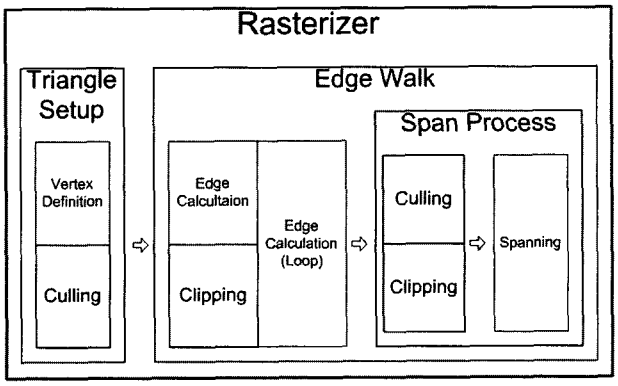


그림 4. 제안하는 래스터라이져 파이프라인
Fig. 4. Proposed Rasterizer Pipeline.

에 관한 프러스텀 컬링 1과 뒷면을 향하는 삼각형에 관한 백 페이스 컬링 처리를 동시에 수행한다. 에지 워크는 Y축을 기준으로 하여 연산하므로 Y축 클리핑을 수행한다. 스팬 프로세서는 Z축과 X축을 기준으로 연산하므로 프러스텀 컬링 2 (Z축 클리핑), X축 클리핑을 동시에 수행한다. 제안하는 래스터라이져는 트라이앵글 셋업, 에지워크, 스팬 프로세스 처리 과정에 컬링과 클리핑을 포함하였다.[5~6][10~11]

3. 제안하는 래스터라이져 설계

본 논문에서 제안하는 래스터라이져는 스캔 라인 (Scan Line) 알고리즘을 사용하였고, 고라우드 쉐이딩 (Gouraud Shading) 효과를 지원한다. 연산에 사용한 덧셈기, 곱셈기, 역수기는 데이터의 복잡도를 고려하여 24 비트 부동소수점 데이터 형식과 16 비트 고정소수점 데이터 형식을 사용하였다. 모델 데이터의 좌표는 복잡도가 높으므로 24 비트 부동소수점 데이터 형식을 사용하였다. 모델 데이터의 색상은 좌표에 비해 복잡도가 낮으므로 16 비트 고정소수점 데이터 형식을 사용하였다.

[그림 4]의 제안하는 래스터라이져는 크게 트라이앵글 세업, 에지 워크, 스팬 프로세스 과정으로 구분한다. 트라이앵글 셋업은 삼각형의 세 개 정점을 정의한다. 세부적으로 버텍스 정의, 백 페이스 컬링, 프러스텀 컬링 1 과정으로 나뉜다. 입력된 세 개의 정점을 Y 축을 기준으로 위치를 정하는 단계가 버텍스 정의이다. 입력된 세 개의 정점이 다른 면에 가려 화면에 보이지 않게 되는 것을 처리하는 단계가 백 페이스 컬링 과정이다. 입력된 세 개의 정점이 화면 영역에서 완전히 벗어나 외부에 있을 때, 이를 구분하는 것이 프러스텀 컬링 1 이다. 에지 워크는 삼각형의 각 변의 정점마다 좌표와 색상을 계산한다. Y축 클리핑, 에지 계산 과정을 거치고, 스팬 프로세서 과정을 포함한다. 화면 영역에서 Y 축을 기준으로 하여 정점의 좌표가 화면 영역을 벗어난 경우에 Y축 클리핑 처리를 한다. 각 에지의 좌표와 색상을 계산하는 과정이 에지 계산 과정이다. 스팬 프로세스는 에지 워크 과정에서 계산된 정점의 좌표와 색상 값으로 주사선 하나를 형성하고, 스패닝하여 색상을 보간한다.[5~6][10~12]

가. 트라이앵글 셋업

[그림 5]의 트라이앵글 셋업 단계 내부에 백 페이스 컬링, 프러스텀 컬링을 포함하며 컬링 처리에 별도 수행 시간의 소요가 없다. 트라이앵글 셋업은 [그림 5]와

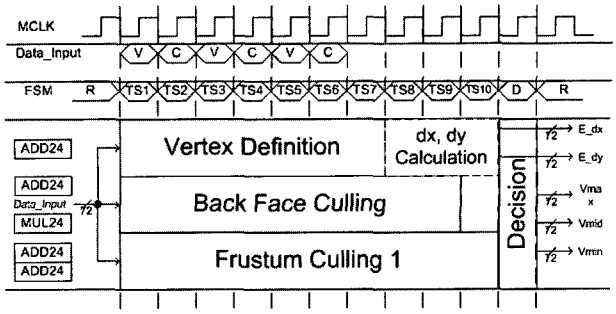


그림 5. 트라이앵글 셋업 단계
Fig. 5. Triangle Setup Stage.

같이 버텍스 정의, 백 페이스 컬링, 프러스텀 컬링 1 과정을 동시에 진행한다. 버텍스 정의는 입력된 세 개의 정점이 Y축 기준으로 Vmax, Vmid, Vmin이라 정의하는 과정이다. 백 페이스 컬링은 입력된 세 개의 정점이 Z축 기준으로 뒷면을 향하는 여부에 따라 에지 워크 과정으로 정점의 전송 여부를 결정한다. 프러스텀 컬링 1 과정은 입력된 세 개의 정점 모두가 화면 영역 외에 존재하면, 에지 워크 과정으로 정점의 정보를 전송하지 않게 한다.

좌표는 24 비트 부동 소수점 형식을 따른다. 좌표 연산은 24 bit 부동소수점 가/감산 연산기와 승산 연산기를 사용한다. 색상 연산은 따로 하지 않는다.

버텍스 정의는 7 클럭, 백 페이스 컬링은 9 클럭, 프러스텀 컬링 1은 10 클럭의 수행시간을 소모하다. x의 기울기, y의 기울기를 3 클럭의 수행시간을 소모한다. 컬링 여부를 판단하는 1 클럭을 포함하여 트라이앵글 셋업의 전체 수행 시간은 11 클럭이다.

(1) 백 페이스 컬링

백 페이스 컬링 처리는 화면에 표시되는 정삼각형의 세 정점의 정보만 유효 판정을 하고, 점선으로 형성되는 삼각형의 세 정점은 유효 판정을 하지 않는다.

세 정점의 x, y 좌표의 값으로 유효 판정을 할 수 있다. 수식 (1)을 정리한 것이 수식 (2)이다. 수식 (2)로 세 정점이 형성하는 삼각형이 화면 앞을 향하는 삼각형인지 화면 뒤를 향하는 삼각형인지 구분한다. 수식 결과 값의 부호가 음수이면 유효 판단을 하고, 양수이면 백 페이스 컬링한다.

$$(Vmid_x - Vmax_x) \times (Vmin_y - Vmax_y) - (Vmin_x - Vmax_x) \times (Vmid_y - Vmax_y) \quad (1)$$

$$(Etop_{dx} \times Emaj_{dy}) - (Emaj_{dx} \times Etop_{dy}) \quad (2)$$

(2) 프러스텀 컬링 1

프러스텀 컬링 1은 삼각형의 화면 영역 위치 여부에 대한 유효 판정하여 유효한 삼각형을 에지 워크 처리하고, 유효하지 않은 삼각형은 에지 워크 처리를 하지 않는다. 각 X, Y, Z축을 기준으로 하여 화면 영역을 벗어난 삼각형의 유효 판정은 X, Y, Z축의 좌표와 삼각형의 세 정점의 감산 연산으로 구분한다. 감산 연산의 서로가 값의 양수/음수인지 구분하여 유효 판정을 한다.

화면 영역을 완전히 벗어난 삼각형을 프러스텀 컬링 1에서 처리하며, Z축이 화면 영역에 걸쳐 있는 삼각형은 프러스텀 컬링 2에서 처리한다. 수식 (3)~(8)을 이용하여 유효 판단을 한다.

$$SC_{SY} - Vmax_y(Vmid_y),(Vmin_y) \quad (3)$$

$$SC_{EY} - Vmax_y(Vmid_y),(Vmin_y) \quad (4)$$

$$SC_{SX} - Vmax_x(Ymid_x),(Vmin_x) \quad (5)$$

$$SC_{EX} - Vmax_x(Ymid_x),(Vmin_x) \quad (6)$$

$$SC_{SZ} - Vmax_z(Ymid_z),(Vmin_z) \quad (7)$$

$$SC_{EZ} - Vmax_z(Ymid_z),(Vmin_z) \quad (8)$$

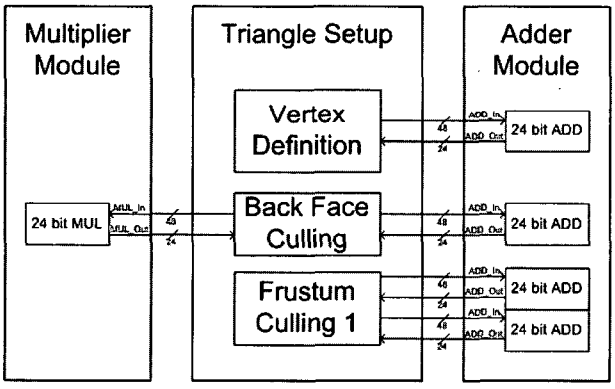


그림 6. 연산기
Fig. 6. Arithmetic Unit.

(3) 연산기 사용 방법

24 비트 부동소수점 연산기는 고정소수점 연산기에 비해 상대적으로 크기가 크므로 연산기의 사용을 적절히 조정한다. 가/감산, 승산 연산이 발생할 때마다 각각의 연산 횟수에 따라 연산기를 사용하면 유닛의 면적이 증가한다. 연산 횟수에 따라 사용하지 않고, [그림 5]와 같이 24 비트 부동소수점 승산기는 1개, 24 비트 부동소수점 가/감산기는 4개를 파이프라인에 맞게 연산한다.

나. 에지 워크

[그림 6]은 세 개의 정점 Vmax, Vmid, Vmin을 삼각형으로 형성하고, 에지를 계산하는 것이 에지 워크 과정이다. 화면 영역에 걸쳐 있는 삼각형이 발생할 경우에는 클리핑 처리를 통해 화면 영역 안의 색상만 그리도록 처리한다. 에지 워크 과정의 처리 순서는 [그림 6]과 같이 Y축 클리핑, 에지 계산, 스팬 프로세스 과정으로 이어진다. 스팬 프로세스 과정이 끝나면, 에지 계산을 하여 스팬한 주사선의 첫, 끝 정점을 계산한다. 이것으로 다시 스팬 프로세서 과정을 수행한다.

Y축 클리핑 과정에서 Y축 기준으로 삼각형이 화면 영역과 영역 바깥에 걸쳐 있는 경우를 처리한다. Y축 클리핑을 하게 되면, 좌표와 색상 보정을 한다. 화면 영역 내로 좌표와 색상을 수정하는 것이다. 스팬 프로세스 과정과 에지 계산 과정을 주고받는 처리 구조이므로 이를 제어하고, 하나의 삼각형을 모두 처리한 후 종료 정보를 트라이앵글 셋업 과정으로 보낸다. 좌표는 24 비트 부동소수점 형식을 따른다. 좌표 연산은 24 비트 부동소수점 가/감산기, 승산기, 역수기를 사용한다. 색상은 16 비트 고정 소수점 형식을 따른다. 색상 연산은 16 비트 고정 소수점 가/감산 연산기, 승산 연산기를 사용한다.

Vmid 방향 정의와 Z, R, G, B의 기울기 계산은 2 클럭, 일반 에지 계산은 7 클럭의 수행시간이 필요하다. 에지 워크의 전체 수행 시간은 9~12 클럭이다. Y축 클리핑을 처리할 때, 별도의 수행 시간이 3 클럭 필요하다. Y축 클리핑 여부에 따라 일반 에지 계산과 Y축 클리핑 연산을 선택한다.

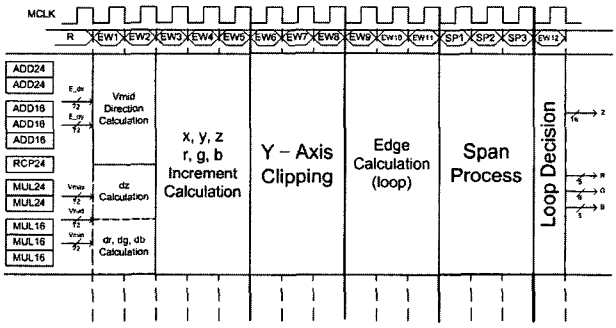


그림 7. 에지 워크 과정 단계
Fig. 7. Edge Walk Stage.

(1) Y축 클리핑

Y축 클리핑 과정은 크게 3 가지 경우로 나뉜다. 첫 번째는 Vmax_y, Vmid_y가 화면 영역 위에 존재하는 경우이다. 두 번째는 Vmax_y만 화면 영역 위에 존재하

는 경우이다. 세 번째는 Vmin_y가 화면 영역 아래에 존재하거나 Vmid_y, Vmid_y가 화면 영역 아래에 존재하는 경우이다. 보이는 화면 영역 내의 삼각형 부분만 처리가기 위해 클리핑을 하고, 좌표와 색상 보정 처리를 한다. 세 번째 경우는 좌표와 색상 보정을 하지 않는다. 화면 영역 아래까지만 처리를 하고 화면 영역을 벗어난 시점부터는 더 이상의 처리를 하지 않음으로써 클리핑을 구현한다. 첫 번째 경우는 수식 (9)~(20)에 따라 좌표와 색상을 보정한다. 두 번째 경우는 수식 (9)~(14), (21)~(20)에 따라 좌표와 색상을 보정한다.

$$V_{SY} = SC_{SY} \tag{9}$$

$$V_{SX} = Vmax_x + \left(\frac{Emaj_{dx}}{Emaj_{dy}}\right) \times (SC_{SY} - Vmax_y) \tag{10}$$

$$V_{SZ} = Vmax_z + \left(\frac{Emaj_{dz}}{Emaj_{dy}}\right) \times (SC_{SY} - Vmax_y) \tag{11}$$

$$V_{SR} = Vmax_r + \left(\frac{Emaj_{dr}}{Emaj_{dy}}\right) \times (SC_{SY} - Vmax_y) \tag{12}$$

$$V_{SG} = Vmax_g + \left(\frac{Emaj_{dg}}{Emaj_{dy}}\right) \times (SC_{SY} - Vmax_y) \tag{13}$$

$$V_{SB} = Vmax_b + \left(\frac{Emaj_{db}}{Emaj_{dy}}\right) \times (SC_{SY} - Vmax_y) \tag{14}$$

$$V_{EY} = SC_{SY} \tag{15}$$

$$V_{EX} = Vmid_x + \left(\frac{Etop_{dx}}{Etop_{dy}}\right) \times (SC_{SY} - Vmin_y) \tag{16}$$

$$V_{EZ} = Vmid_z + \left(\frac{Etop_{dz}}{Etop_{dy}}\right) \times (SC_{SY} - Vmin_y) \tag{17}$$

$$V_{ER} = Vmid_r + \left(\frac{Etop_{dr}}{Etop_{dy}}\right) \times (SC_{SY} - Vmin_y) \tag{18}$$

$$V_{EG} = Vmid_g + \left(\frac{Etop_{dg}}{Etop_{dy}}\right) \times (SC_{SY} - Vmin_y) \tag{19}$$

$$V_{EB} = Vmid_b + \left(\frac{Etop_{db}}{Etop_{dy}}\right) \times (SC_{SY} - Vmin_y) \tag{20}$$

$$V_{EY} = SC_{SY} \tag{21}$$

$$V_{EX} = Vmax_x + \left(\frac{Etop_{dx}}{Etop_{dy}}\right) \times (SC_{SY} - Vmin_y) \tag{22}$$

$$V_{EZ} = Vmax_z + \left(\frac{Etop_{dz}}{Etop_{dy}}\right) \times (SC_{SY} - Vmin_y) \tag{23}$$

$$V_{ER} = Vmax_r + \left(\frac{Etop_{dr}}{Etop_{dy}}\right) \times (SC_{SY} - Vmin_y) \tag{24}$$

$$V_{EG} = Vmax_g + \left(\frac{Etop_{dg}}{Etop_{dy}}\right) \times (SC_{SY} - Vmin_y) \tag{25}$$

$$V_{EB} = Vmax_b + \left(\frac{Etop_{db}}{Etop_{dy}}\right) \times (SC_{SY} - Vmin_y) \tag{26}$$

(2) 연산기 사용 방법

에지 워크 과정은 24 비트 부동소수점 연산기와 16 비트 고정소수점 연산기를 복합적으로 사용하므로 연산기의 사용을 적절히 조정한다. 가/감산, 승산 연산이 발생할 때마다 각 연산 횟수에 따라 연산기를 사용하면 유닛의 면적이 증가한다. 연산 횟수에 따라 사용하지 않고, [그림 7]과 같이 24 비트 부동소수점 승산기는 2개, 24 비트 부동소수점 가/감산기는 2개, 24 비트 역수기는 1개, 16 비트 고정소수점 승산기는 3개, 16 비트 고정소수점 가/감산기는 3개를 파이프라인에 맞게 연산한다.

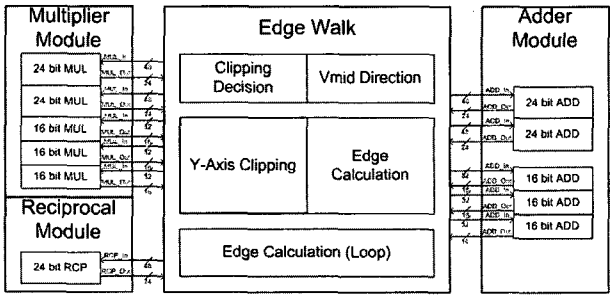


그림 8. 연산기
Fig. 8. Arithmetic Unit.

다. 스팬 프로세스

[그림 8]은 스팬 프로세스 단계 내부에 프러스텀 컬링 2와 X축 클리핑을 포함하여 컬링, 클리핑 처리에 별도 수행 시간의 소요가 없다.

[그림 8]의 스팬 프로세스 과정은 프러스텀 컬링 2,

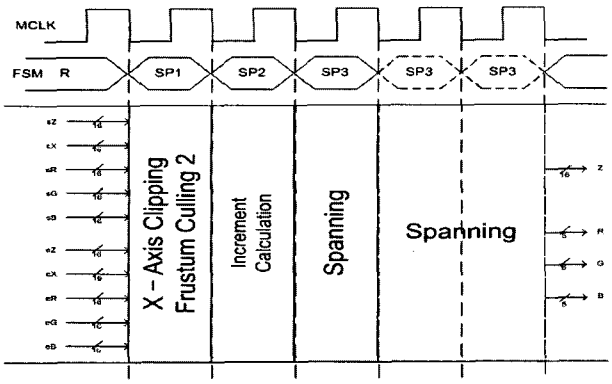


그림 9. 스팬 프로세스 단계
Fig. 9. Span Process Stage.

X축 클리핑, 스패닝으로 구분한다. X축을 기준으로 화면 영역 밖의 좌표를 화면 영역 내로 좌표와 색상 보정을 한다. 스패닝 과정은 두 정점 사이의 주사선 내부를 색상 정보로 채워 나가는 과정이다. 스팬 프로세스 과정은 모두 16 비트 고정 정수 형식을 따른다. 입력된 좌표와 색상 값은 16 비트 정수형으로 변환하여 연산한다. 설계한 스패닝 과정은 1 클럭에 1 픽셀을 계산하도록 고안하였다.

(1) 프러스텀 컬링 2

프러스텀 컬링 2에서는 화면 영역 밖에 있는 부분을 제거하여 좌표와 색상 보정을 한다. 프러스텀 컬링 1에서는 화면 영역 밖에 삼각형이 형성이 되므로, 형성되는 경우를 모두 계산하였다.

V_sz 값이 화면 영역 시작 Z 보다 작은 경우는 좌표와 색상 보정을 한다. 좌표와 색상 보정은 수식 (27)~(30)을 이용 하여 Z축에 관해 화면에 보이는 영역까지 스팬 처리한다.

$$V_{SZ} = SC_{SZ} \tag{27}$$

$$V_{SR} = V_{SR} + SP_{dr} \times \left\{ \frac{(SC_{SZ} - V_{SZ})}{SP_{dz}} \right\} \tag{28}$$

$$V_{SG} = V_{SG} + SP_{dg} \times \left\{ \frac{(SC_{SZ} - V_{SZ})}{SP_{dz}} \right\} \tag{29}$$

$$V_{SB} = V_{SB} + SP_{db} \times \left\{ \frac{(SC_{SZ} - V_{SZ})}{SP_{dz}} \right\} \tag{30}$$

(2) X축 클리핑

프러스텀 컬링 2와 같은 방법을 사용하여 화면 영역을 벗어난 부분은 제거하여 연산 낭비를 줄이고자 한다. V_sx 값이 화면 영역 시작 X 보다 작은 경우는 좌표와 색상 보정을 한다. 좌표와 색상 보정은 수식 (31)~(34)를 이용 하여 X축에 관해 화면에 보이는 영역까지 스팬 처리한다.

$$V_{SX} = SC_{SX} + V_{SX} \tag{31}$$

$$V_{SR} = V_{SR} + SP_{dr} \times \left\{ \frac{(SC_{SX} - V_{SX})}{SP_{dx}} \right\} \tag{32}$$

$$V_{SG} = V_{SG} + SP_{dg} \times \left\{ \frac{(SC_{SX} - V_{SX})}{SP_{dx}} \right\} \tag{33}$$

$$V_{SB} = V_{SB} + SP_{db} \times \left\{ \frac{(SC_{SX} - V_{SX})}{SP_{dx}} \right\} \tag{34}$$

4. 제안하는 래스터라이져 성능 비교

본 논문에서 제안한 래스터라이져와 타 래스터라이져의 성능을 [표 1]에 비교하였다. 비교 항목은 최대 동작 주파수, 최대 해상도, 최대 지원 색상, 깊이 값, 처리 시간, 평균 처리 속도이며, 비교 대상은 "휴대형기기에

표 1. 성능 비교
Table 1. Performance Comparison.

		1	2	제안하는 래스터라이져
최대 동작주파수			36.4MHz	52MHz
최대 해상도		.	800*600	65536*65536 (240*320)
최대 지원 색상		.	32-bit (RGBA)	24-bit (RGB)
깊이 값		.	16-bit	16-bit
사용 FPGA (Gate)		.	Xilinx VITEX 2 600만	Xilinx VITEX 2 600만
처리 시간 (Clock)	트라이앵글 셋업	15	42	11
	에지 워크	6	19	9 ~ 12
	스팬 프로세스	6	9	3
	처리시간 합계	27	70	23 ~ 26
평균 처리 속도		40M pixel/s	48M pixel/s	52M pixel/s

1 : 병렬 구조를 가진 3차원 그래픽 렌더링 하드웨어 연구
2 : 휴대형기기에 적합한 내장형 3차원 그래픽 렌더링 처리기 설계

적합한 내장형 3차원 그래픽 렌더링 처리기 설계" - 우현재(연세대), "병렬 구조를 가진 3차원 그래픽 렌더링 하드웨어 연구" - 정종철(연세대)의 석사 학위 논문이다. 두 논문의 래스터라이져는 컬링, 클리핑 유닛을 포함하지 않은 구조이며, 제안하는 래스터라이져는 컬링, 클리핑 유닛을 포함한 구조이다. [표 1]과 같이 제안하는 래스터라이져의 첫 픽셀이 출력되는 처리 시간은 일반 삼각형 처리시에 23 클럭, Y축 클리핑 처리시에 26 클럭으로 타 래스터라이져에 비해 컬링과 클리핑을 포함하고도 처리 시간이 향상되었고, 초당 52M pixel 처리로 평균 처리 속도도 향상되었다.[7~9]

Ⅲ. 실 험

본 논문은 24 비트 부동소수점 데이터 형식, 16 비트 고정소수점 데이터 형식을 처리하는 연산기를 설계하고, 이를 이용하여 래스터라이져를 설계하였다. 일반적으로 설계 검증은 설계한 유닛에 대한 로직 시뮬레이션으로 회로 기능을 검사하고, 설계한 유닛을 FPGA에 구현하여 검증환경을 구성한다. 제안한 래스터라이져의 설계 검증은 연산기, 래스터라이져 유닛, 소프트웨어, 하드웨어로 구분하여 진행하였다. 로직 시뮬레이션을 통해 회로검증을 하고, 소프트웨어 검증을 통해 래스터라이져 유닛의 정상 동작을 확인하였으며 하드웨어 검증을 통해 [그림 11]과 같이 LCD에 출력하였다.

가/감산기, 곱셈기, 역수기 등의 연산기와 제안한 래스터라이져는 Mentor사의 ModelSim을 이용하여 로직 시뮬레이션으로 회로 기능을 확인하였다. [그림 9]는 래스터라이져 유닛의 로직 시뮬레이션 결과이다.

소프트웨어 검증은 ModelSim과 Visual C++ .NET로 구성하였다. 약 2000개 정도의 폴리곤을 갖는 모델들을 사용하였다. ModelSim을 통해 모델의 데이터를 HDL로

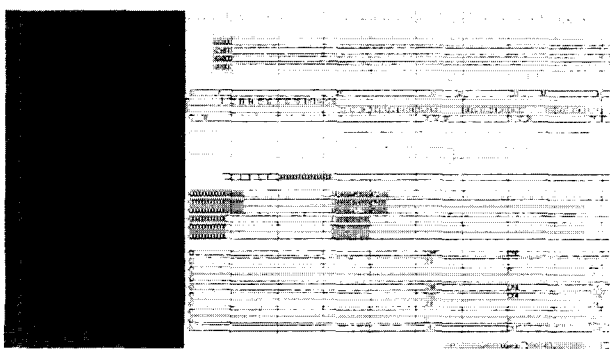


그림 9. 로직 시뮬레이션
Fig. 9. Logic Simulation.

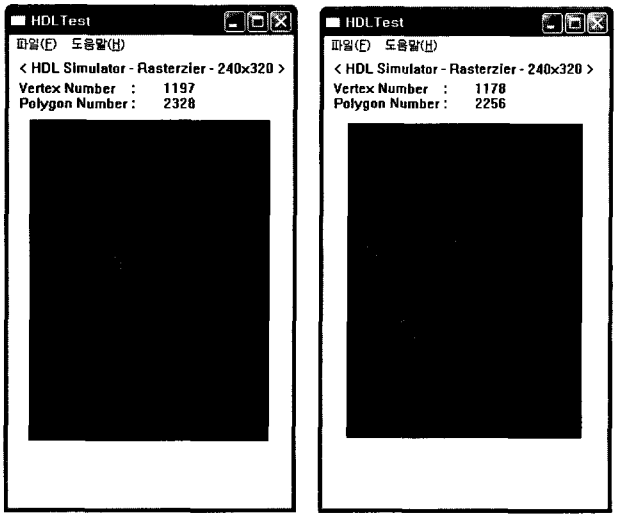


그림 10. 소프트웨어 데모
Fig. 10. Software Demo.

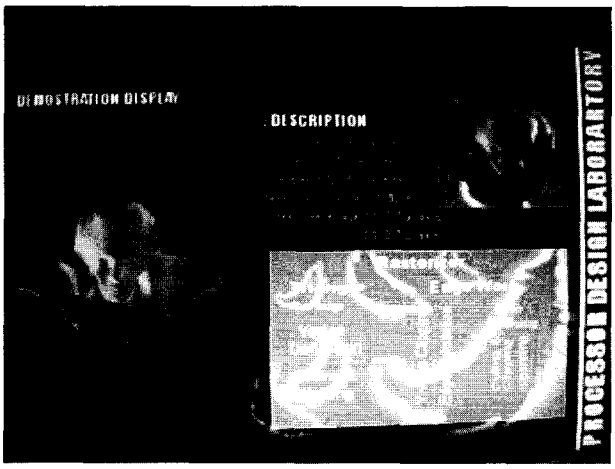


그림 11. 하드웨어 데모
Fig. 11. Hardware Demo.

Unit		Maximum Speed	Clock
FPGA	TOP	52 MHz	23 ~26
	Triangle Setup	70 MHz	11
	Edge Walk	52 MHz	9 ~ 12
	Span Process	106 MHz	3
First Pixel Output Latency		23~26 Clock	
Throughput		1 Pixel / 1 Clock	
Maximum Resolution		65536*65536 (240*320)	
Color(RGB)		24-bit (16-bit)	
Depth(Z)		16-bit	
FPGA		Xilinx VITEX 2 - 600M Gate	

그림 12. 래스터라이져 성능
Fig. 12. Rasterizer Performance.

래스터라이져 처리한다. ModelSim 처리 후의 색상 값을 C 환경에서 처리하여 [그림 10]의 검증환경을 구성하였다.[13~14]

SoC 검증 보드의 600만 게이트 FPGA에 래스터라이져를 구성하였다. SoC 검증 보드는 ARM 기반의 플랫폼으로 AMBA BUS를 통해 FPGA와 데이터 전송이 가능한 환경을 제공한다. 모델 데이터를 AMBA BUS로 전송하여 FPGA의 래스터라이져에서 처리 하고, [그림 11]과 같이 LCD 모니터에 출력하였다.

Ⅳ. 결 론

제안한 래스터라이져는 고라우드 쉐이딩을 지원한다. [그림 11]은 컬링과 클리핑을 포함한 각 유닛의 성능 수치이다. Y축 클리핑 처리 시에 3 클럭의 별도 추가 수행시간을 소요하였다. 제안한 래스터라이져는 최대 52 MHz 주파수에 동작하며, 정점 3개의 입력 후 23 클럭에 첫 번째 픽셀을 출력한다. Y축 클리핑 처리 시에는 정점 3개의 입력 후 26 클럭에 첫 번째 픽셀을 출력한다. 이후 1 클럭마다 1 픽셀을 출력하여 초당 52M pixels의 평균 처리 속도 성능을 나타낸다. 최대 해상도는 $2^{16} \times 2^{16}$으로 65536X65536 크기 표현이 가능하다. 색상은 RGB가 각각 8bit 으로 24 bit, 깊이는 16bit 지원 가능하다. 특히, 본연구에서는 저면적,저전력을 필요로하는 모바일 기기에 적합하도록 기존의 32비트 연산방식을 24비트 및 16비트 연산방식으로 해상도 저하없이 최적화 하였다.

참 고 문 헌

[1] Richard S, "OpenGL SUPERBIBLE", Third Edition, 2004.

[2] Tomas, Eric, "Real-Time Rendering", Second Edition, 2002.

[3] Foley, van Dam, "Computer Graphics Principles and Practice", Second Edition, 2003.

[4] Samuel R. Russ, "3D Computer Graphics: a Mathematical Introduction with OpenGl", Cambridge Univ. press, 2001.

[5] Anders Kugler, "The Setup for Triangle Rasterization", 11th EUROGRAPHICS Workshop on Computer Graphics Hardware, August 26-27, Poitiers, France, 1996.

[6] Joel McCormack, Robert McNamara, "Tiled Polygon Traversal using Half-Plane Edge Functions", Proceedings of the 2000 EUROGRAPHICS/SIGGRAPH Workshop on Graphics Hardware, ACM Press, New York, August 2000. pp.15-21

[7] Cheol-Ho Jeong, "Design of an Effective Control and Execution Method for Geometry Engines and Rasterizers within Embedded 3D Graphics Accelerators", Yonsei Univ., PhD Thesis, 2003.12.

[8] 정종철, "병렬 구조를 가진 3차원 그래픽 렌더링 하드웨어 연구", 연세대 대학원, 2001.12.

[9] 우현재, "휴대형기기에 적합한 내장형 3차원 그래픽 렌더링 처리기 설계", 연세대 대학원, 2004.1.

[10] YOU-DONG LIANG, BRIAN A. BARSKY, "A New Concept and Method for Line Clipping" ACM Transaction on Graphics, p1-p22, Vol 3, No 1, January 1984.

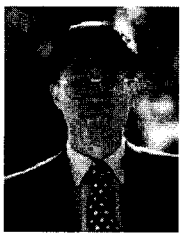
[11] 고찬, "효과적인 클리핑 알고리즘에 관한 고찰", 서울산업대학교논문집 제21집, p131-137, 1985. 8.

[12] IEEE std 754-1985, "IEEE standard for binary floating-point arithmetic", IEEE, 1985.

[13] ModelSIm Command Reference version 6.0c - Mentor Graphics

[14] Microsoft MSDN Library -http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/functions/shellexecute.asp

저 자 소 개



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

구 용 서(평생회원)
1981년 서강대학교 전자공학과 학사.
1983년 서강대학교 전자공학과 석사.
1993년 서강대학교 전자공학과 박사.
1983년~1993년 한국전자통신연구원 선임 연구원
1993년~현재 서경대학교 전자공학과 부교수.
<주관심분야 : Smart Power IC 설계, 나노 ESD 보호회로 설계>