논문 2008-45SD-8-10

효율적인 클리핑 기능을 갖는 3차원 그래픽 파이프라인 구조

(A 3D graphic pipelines with an efficient clipping algorithm)

이 찬 호*

(Chanho Lee)

요 약

최근 모바일 기기에 3차원 그래픽 디지털 콘텐츠들이 증가함에 따라 휴대용 기기에 적합한 3차원 그래픽 가속기의 연구와 설계는 점점 중요한 이슈가 되고 있다. 본 논문에서는 저전력 3차원 그래픽 파이프라인에 적합한 효율적인 클리핑 구조를 제안한다. 많은 연산 사이클과 연산기를 필요로 하는 클리핑 연산을 두 단계로 나누어서 기하변환 엔진에서는 컬링 정렬(cull and sort) 유닛으로 구현하고, 실질적인 클리핑은 스캔 변환(scan conversion)에서 구현한다. 즉, 스캔 변환 처리기를 구성하고 있는 변처리 (edge walk) 유닛에서 Y축 클리핑을 함께 수행하고 스팬처리 (span processing) 유닛에서 X축과 Z축 클리핑을 함께 수행한다. 제안하는 기하 변환 엔진의 컬링 정렬 유닛은 기존 클리핑 유닛에 비해 면적과 동작 사이클이 크게 줄었고 스캔 변환 처리기의 면적은 거의 증가하지 않아 전반적으로 동작 속도 및 동작 효율을 높였다. 제안하는 클리핑 구조를 적용한 3차원 그래픽 가속기는 Verilog-HDL을 이용하여 설계하고 FPGA를 이용하여 검증하였다.

Abstract

Recently, portable devices which require small area and low power consumption employ applications using 3D graphics such as 3D games and 3D graphical user interfaces. We propose an efficient clipping engine algorithm which is suitable in 3D graphics pipeline. The clipping operation is divided into two steps: one is the selection process in the transformation engine and the other is the pixel clipping process in the scan conversion unit. The clipping operation is possible with addition of simple comparator. The clipping for the Y-axis is achieved in the edge walk stage and that for the X and Z-axis is performed in the span processing. The proposed clipping algorithm reduces the operation cycles and the area of of 3D graphics pipelines. We designed a 3D graphics pipeline with the proposed clipping algorithm using Verilog-HDL and verifies the operation using an FPGA.

Keywords: 3D graphic processor, Rendering engine, Clipping, Edge walk, Span processing, Scan conversion

I. 서 론

3차원 그래픽 연산이란 가상 카메라, 3차원 객체, 광원 조명처리 모델, 텍스쳐 등이 주어졌을 때 그것으로부터 2차원 이미지를 만들어 내는 것, 다시 말해서 렌더링 하는 것이다[1]. 과거에는 컴퓨터 그래픽스 기술을 적용하기 위해서는 고가의 그래픽 워크스테이션과 그래픽 소프트웨어가 필요하였다. 현재는 PC 환경에서도 3차원 그래픽 가속기가 기본적으로 장착되고 있어서 실시간 3차원 그래픽스와 애니메이션이 가능하게 되었다[2]. 최근 그래픽 하드웨어의 발달로 인하여 많은 디지털 콘텐츠들이 3차원 그래픽을 기반으로 제작되고 이들이 2차원 그래픽 기반의 콘텐츠에 비하여 좋은 반응을 보임에 따라 많은 사용자들을 확보하고 있는 휴대폰, PMP, PDA등의 모바일 기기에서도 3차원 그래픽 시스템에 대한 관심이 증가하고 있다[3]. 3차원 그래픽 처리 과정은 2차원 그래픽 처리에 비하여 많은 양의 연산을 필요로 하고 그 연산 또한 복잡하다. 기존 워크스테이션이나 일반 PC의 3차원 그래픽 프로세서는 고성능의 CPU와 무한한 전력을 제공받을 수 있기 때문에 그러한 문제점들에 대해서는 크게 고려하지 않고 있었으나 유한한 자원과 작은 크기를 특징으로 하는 모바일 기기에

* 정회원, 숭실대학교 정보통신전자공학부
(School of Electronic Engr., Soongsil University)
※ 본 연구는 숭실대학교 교내연구비 지원으로 이루어졌음
접수일자: 2008년3월24일, 수정완료일: 2008년7월21일

적용 가능한 마이크로프로세서들은 제한된 연산 능력을 가지고 있으며, 3차원 그래픽 연산 이외에도 다른 기능들을 동시에 서비스 해줄 수 있어야 한다. 따라서 기존 워크스테이션이나 일반 PC에서 사용하고 있는 3차원 그래픽 기술을 그대로 가져올 수 없다[1]. 따라서 이러한 내장형 프로세서의 부담을 줄여주며, 전력을 적게 소모하는 효율적인 3차원 그래픽 프로세서가 필요하다.

기하 변환 엔진(geometry engine)에서 처리하는 기존의 클리핑(clipping) 연산은 클리핑 영역 밖에 존재하는 정점과 클리핑 영역 안에 존재하는 정점을 이용하여 새로운 정점을 생성하는 작업을 함으로써 클리핑 영역 밖에 있는 정점을 제어하는 일을 수행하기 때문에 많은 연산 사이클과 연산기를 필요로 하는 작업이다.

따라서 본 논문에서는 많은 연산 사이클과 연산기를 필요로 하는 클리핑 연산을 두 단계로 나누어서 기하변환 엔진에서는 단순히 삼각형 전체가 클리핑 영역 밖에 존재하는지를 판단하도록 하여 하나의 정점이라도 클리핑 영역 안에 걸쳐있는 경우에만 다음 단계로 삼각형을 보내도록 구현하고, 가장자리에 걸쳐있는 삼각형에 대한 처리는 스캔 변환(scan conversion) 단계에서 구현하도록 제안한다. 이를 위해 삼각형 구성을 스캔 변환 단계가 아닌 기하 변환 엔진에서 수행하고 백 페이스 컬링(Back-face culling)도 동시에 진행한다. 스캔변환 단계에서는 픽셀 단위 연산에서 간단한 비교 연산만 추가하면 원하는 기능을 수행하므로 면적의 증가는 거의 없다. 제안하는 구조에 따라 3차원 그래픽 엔진을 파이프라인이 가능하도록 설계하여 동작을 검증하였다.

II. 새로운 클리핑 구조

1. 3차원 그래픽 하드웨어의 구조

가 3차원 그래픽 처리과정

기본적인 3차원 그래픽 연산은 그림 1과 같은 순서로 진행되며 크게 두 가지 과정으로 분류할 수 있다. 전반부는 카메라와 같은 시점의 위치와 방향을 결정하여 모든 삼각형들을 그 시점에 의한 시점 좌표계로 변환하는 시점 변환(viewing transform) 단계와 현실세계의 여러 빛의 요소를 고려하여 물체의 명암을 결정하는 조명 연산(lighting) 단계, 그리고 실제로 화면에 보이는 영역인 뷰 볼륨(view volume) 밖에 있는 삼각형들을 잘라내어 불필요한 연산을 줄여주기 위한 클리핑 단계, 2차원 화면 이미지로 옮기기 위한 투영 변환

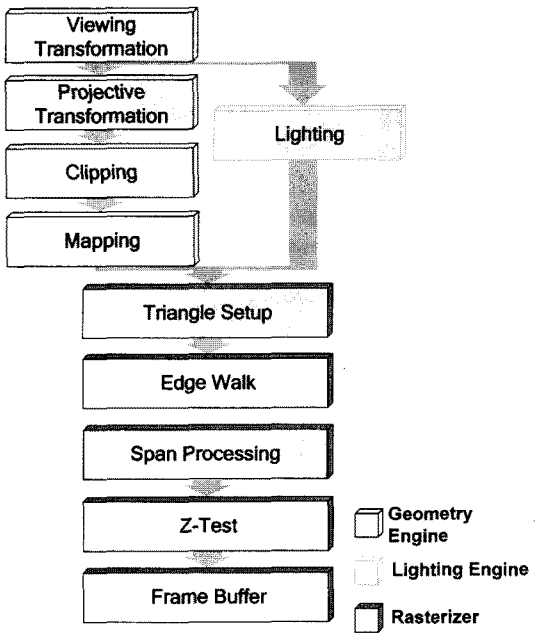

그림 1. 3차원 그래픽 연산 파이프라인
Fig. 1. 3D graphic processing pipeline.

(projective transform) 단계로 이루어져 있는 기하 연산부이다.

후반부는 이러한 기하 변환 연산을 통해 생성된 데이터를 기반으로 세 정점을 이용하여 삼각형 내부의 픽셀에 대한 색상 정보를 구하는 스캔 변환 단계, 그림자 처리와 면에 재질감을 넣는 텍스쳐 매핑, 투명도 처리, 안개 처리, 안티 앨리어싱, 깊이 테스트 (Z-Test) 등을 거치는 렌더링 과정이다[4].

나. 기존의 클리핑 알고리즘

클리핑 영역은 관측자가 볼 수 있는 영역 공간으로 이를 벗어나는 물체는 화면상에 표시되지 않는다. 만약 클리핑 영역 밖의 모든 물체에 대해 렌더링을 하게 된다면 불필요한 많은 연산을 해야 한다. 관측자의 시야 영역에서 벗어나 있는 요소들을 그리는 일은 보이지 않는 공간을 그리게 되는 불필요한 작업이기 때문에 클리핑 과정을 통하여 이러한 불필요한 요소들을 사전에 제거한다. 즉, 스캔 변환 과정을 수행하기 전에 시야에서 벗어난 다각형을 미리 제거한다면 래스터화 단계에서의 연산량이 크게 줄어 전체 처리 성능이 향상된다. 따라서 클리핑 연산은 일반적으로 스캔 변환이 일어나기 전인 기하 변환 단계에서 수행된다.

그림 2에 일반적인 클리핑 알고리즘을 보여주는 그림이 나타나 있다. 이를 구현할 수 있는 기존의 클리핑 알고리즘은 여러 가지가 있으나 많이 사용되는 방법으

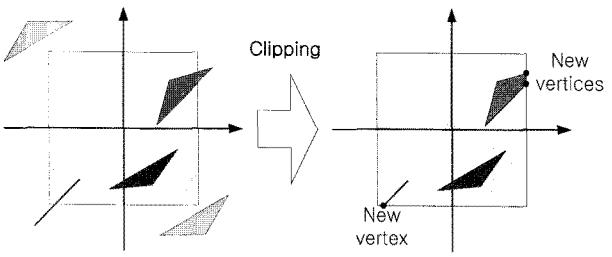

그림 2. 일반적인 클리핑 알고리즘
Fig. 2. General clipping algorithm.

로 Cohen-Sutherland의 알고리즘과[5] Liang-Barsky의 알고리즘을[6] 혼합하여 3차원 형태로 확장한 형태의 알고리즘이 있다[7]. 이 알고리즘에서는 우선 각 좌표의 정보를 통하여 폴리곤의 위치를 판단하여 완전히 제거해야 할 폴리곤인지, 클리핑 영역 안에 있어서 그대로 다음 단계로 넘겨야 할 폴리곤인지, 클리핑 영역에 걸쳐져 있어서 새로운 좌표 계산이 필요한 폴리곤인지를 결정한다. 첫 번째와 두 번째 경우는 간단하지만 세 번째 경우는 클리핑 범위에 걸쳐 있는 삼각형을 분할하여 클리핑 범위 내의 삼각형으로 재구성 하는 과정을 수행하기 때문에 많은 연산을 필요로 하고 이 과정에서 정점의 수가 늘어나기도 한다. 이는 기하 연산 엔진의 파이프라인을 훼손시키면서 렌더링 유닛에 전달해야 하는 정점의 수를 증가시키고 클리핑 연산기를 복잡하게 만드는 원인이 되기도 한다[8]. 또한 클리핑을 기하변환 단계의 마지막 단계에서 진행하는 것이 아니기 때문에 재구성된 삼각형을 남은 기하변환 단계 동안 정점 단위로 연산을 진행하여 이후 삼각형 셋업(Triangle setup) 과정에서 다시 삼각형을 구성하는 오버헤드가 발생한다.

다. 제안하는 클리핑 구조

이러한 문제를 해결하기 위해 스캔 변환 과정에서 클리핑 연산을 수행하도록 하면 간단한 비교기만을 사용하여 원하는 동작을 하도록 할 수 있다. 따라서 그림 3과 같이 클리핑 연산을 컬링 정렬 유닛과 스캔 변환 유닛의 변처리와 스팬처리 유닛으로 분산하여 연산 사이클과 연산기 수를 줄일 수 있다.

(1) 컬링 정렬 (Cull and Sort) 유닛

기존의 클리핑 알고리즘은 클리핑 범위에 걸쳐있는 객체들을 연산하기 위하여 많고 불규칙한 연산량을 필요로 한다. 이를 개선하기 위해 클리핑 연산을 스캔 변환 단계로 보내는 방식이 소개되었다. 그러나 클리핑 연산을 모두 후반부로 보내면 클리핑 영역 밖의 다각형

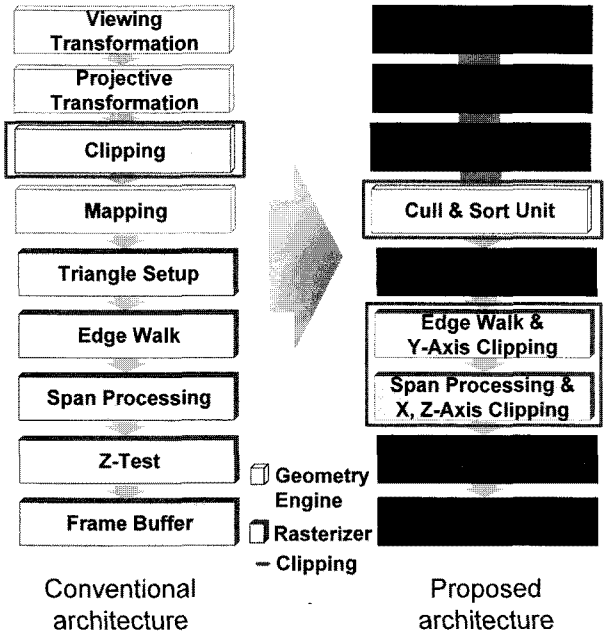

그림 3. 제안된 3차원 그래픽 파이프라인의 클리핑 연산
Fig. 3. Proposed clipping algorithm for fixed 3D graphic pipelines.

에 대해서도 삼각형 셋업과 변처리(edge walk) 작업을 수행해야 하므로 비효율적이다.

따라서 제안하는 알고리즘에서는 기하변환 단계에서 그림 4와 같이 Cohen-Sutherland의 알고리즘에 따라 플래그 정보를 계산하고 그 플래그를 통하여 위치를 판단하는 방식을 적용하여 컬링 정렬 유닛을 구성하였다. 컬링 정렬 유닛으로 삼각형이 완전히 클리핑 영역(또는 view frustum) 밖에 존재하는지 아니면 하나의 정점이라도 내부에 걸쳐져 있는지 여부를 판단하고, 완전히 밖에 있는 삼각형은 제거하여 이후 단계로 전달하지 않는다. 즉, 삼각형의 위치만을 파악할 뿐, 기존의 클리핑 알고리즘과는 달리 클리핑 범위에 걸쳐있는 객체들을 처리하지 않고 스캔 변환 단계로 보낸다. 간단한 비교 연산을 통하여 클리핑 영역에 하나의 정점이라도 걸쳐있는 삼각형은 제거하지 않고, 완전히 클리핑 영역을 벗어난 삼각형들을 제거하므로 연산량과 스캔 변환 유닛으로 보내는 정점의 수를 크게 줄일 수 있다. 그림 4(c)에서 기존의 방식은 클리핑 후 새로운 정점을 생성하여 15개의 정점을 스캔 변환 유닛으로 보내지만 제안된 방식에서는 그림 4(b)와 같이 9개의 정점만을 보낸다.

클리핑 연산을 위해서는 삼각형이 구성되어야 하므로, 기존에 삼각형 셋업 단계에서 진행하는 정점을 정

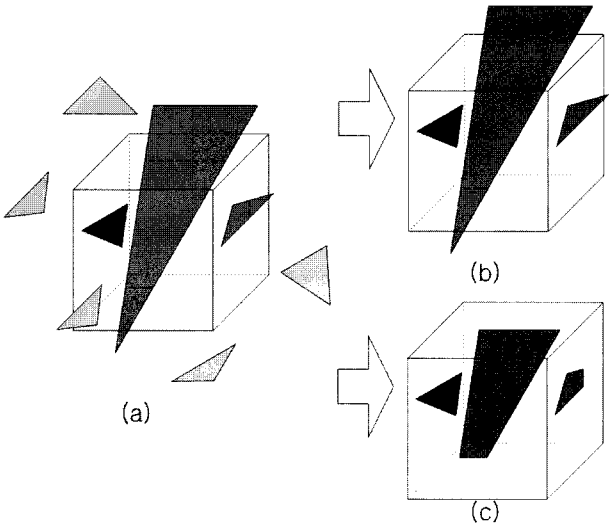

그림 4. 제안된 클리핑 알고리즘과 기존 방식과의 비교. (a) 클링핑 전 (b) 컬링 정렬 유닛 통과 후 (c) 기존 클리핑 방식 결과

Fig. 4. Comparison of the proposed clipping algorithm with the conventional clipping method. (a) Before clipping (b) After the proposed Cull and Sort unit (c) After the conventional clipping unit.

렬하고 삼각형을 만드는 작업을 이곳에서 진행함으로써 삼각형 셋업 유닛의 부담을 줄이고 구조를 단순화시킬 수 있다. 이를 위해 매핑 변환(mapping transformation) 연산을 클리핑 연산 앞으로 옮겨 투영 변환과 동시에 수행하도록 하여 기하변환 유닛을 더욱 단순화 하였다. 이러한 구조적 개선을 통하여 기하 변환 엔진은 한 사이클 만에 삼각형이 완전히 클리핑 범위를 벗어났는지 여부만을 판단함으로써 기하 변환 엔진에서 마지막까지 파이프라인을 유지시키면서 불필요한 연산 사이클과 면적을 크게 줄이는 효과를 얻을 수 있다.

(2) Y축 클리핑을 포함한 변처리 유닛

일반적으로 변처리 유닛에서는 Y축 값을 1씩 증가시키면서 각 변 즉, 스팬의 시작점과 끝점의 색상과 좌표를 결정짓는 일을 수행한다. 이러한 특징을 이용하여 본 논문에서는 변처리 유닛에서 Y축 클리핑을 함께 수행할 수 있도록 하였다.

삼각형 전체가 클리핑 영역을 완전히 벗어난 경우에는 이미 컬링 정렬 유닛에서 제거되었기 때문에 변처리 유닛의 입력으로 들어오는 삼각형은 Y축 클리핑 영역에 걸쳐져 있거나 삼각형 세 정점이 모두 Y축 클리핑 영역 내부에 존재한다. 삼각형이 클리핑 영역 아랫변에 걸쳐있는 경우는 그림 5에 나타난 바와 같이 Y 값이 가장 큰 정점부터 Y를 1씩 감소시키면서 변처리 연산

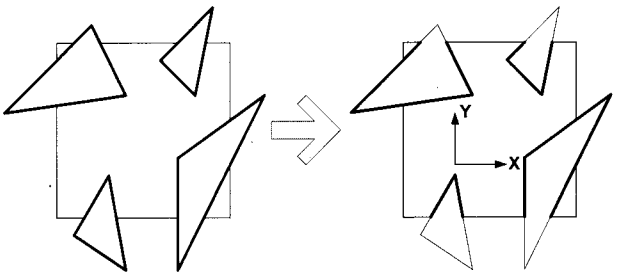

그림 5. Y축 클리핑을 포함한 변처리 유닛의 결과

Fig. 5. Results of edge walk processing including Y-axis clipping.

을 진행하면서 클리핑 영역 아랫변에 도달하게 되면 변처리 연산을 종료한다. 그 외의 경우 즉, 삼각형이 클리핑 영역 윗변에 걸쳐있거나 위 아랫변 모두 걸쳐있거나 삼각형이 완전히 Y축 클리핑 영역 안에 존재하는 경우에는 Y 값이 가장 작은 정점부터 Y를 1씩 증가시키면서 연산을 진행하면서 클리핑 영역 윗변에 도달하거나 Y 값이 가장 큰 정점에 도달하게 되면 변처리 연산을 종료한다.

제안하는 구조에 따르면 클리핑 영역의 위 아랫변 모두 걸쳐져있는 경우에 클리핑 영역 아랫변의 아래쪽으로 벗어난 y 길이만큼의 불필요한 연산을 진행하는 단점이 있지만 실질적으로 사용되는 3차원 물체를 렌더링할 때 그와 같은 경우의 발생 확률이 거의 없기 때문에 큰 문제가 되지 않는다.

이러한 Y축 클리핑을 변처리 유닛에서 처리하기 위하여 추가되는 자원은 Y 축 클리핑 영역을 벗어났는가의 여부만을 판단할 수 있는 비교기와 컨트롤 레지스터 뿐이다. 오히려 기하 변환 단계에서 클리핑을 하였을 때 새로운 정점 생성으로 인해 늘어난 삼각형만큼 변처리를 진행하는 것에 비하여 효율적인 구조이다.

(3) X축과 Z축 클리핑을 포함한 스팬처리 유닛

일반적으로 스팬처리 유닛에서는 X를 1씩 증가시키면서 스팬 내부 픽셀의 색과 좌표를 결정한다. 이러한 특징을 이용하여 스팬처리 유닛에서 X축과 Z축 클리핑을 함께 수행할 수 있는 구조를 제안한다.

클리핑 영역을 완전히 벗어난 삼각형은 이미 컬링 정렬 유닛에서 제거되지만 변처리 유닛을 거치고 나면 Y축을 벗어나는 부분만 처리되므로 그림 6의 ①과 같이 스팬의 시작점이나 끝점이 X축 클리핑 영역을 벗어나는 경우가 발생한다. 스팬처리 유닛에서는 스팬을 변처리 유닛으로부터 입력받으면 우선 스팬의 시작점이 클리핑 영역의 우변보다 큰지 여부와 스팬의 끝점이 클리

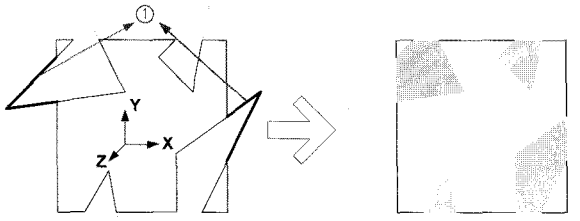

그림 6. X축 클리핑을 포함한 스팬처리 유닛의 결과
Fig. 6. Results of span processing including X- and Z-axis clipping.

핑 영역의 좌변보다 작은지 여부를 판단하여 그에 해당하는 경우 즉, 완전히 클리핑 영역을 벗어난 스팬은 제거한다. 또한 스팬의 시작점과 끝점의 Z 값이 모두 클리핑 영역 밖에 존재하는지 여부를 비교하여 제거한다.

사전에 완전히 제거되는 스팬을 제외하면 스팬처리 유닛에서 처리해야할 스팬은 양 끝점이 X축 클리핑 영역의 안팎으로 걸쳐있거나 스팬의 시작점과 끝점 모두 X축 클리핑 영역 내부에 존재한다. 스팬의 시작점이 X축 클리핑 영역 밖에 존재하는 경우는 스팬의 끝점부터 스팬처리를 시작하면서 (즉, 오른쪽에서 왼쪽으로) X 값이 클리핑 영역 좌변에 도달하게 되면 스팬처리 연산을 종료한다. 그 외의 경우 즉, 스팬의 끝점이 X축 클리핑 영역 밖에 존재하거나 스팬의 시작점과 끝점 모두 X축 클리핑 영역 안에 존재하는 경우에는 스팬의 시작점부터 스팬처리를 시작하면서 (왼쪽에서 오른쪽으로) X 값이 클리핑 영역 우변에 도달하거나 끝점에 도달하게 되면 스팬처리 연산을 종료한다. 이러한 연산 후 스팬처리 유닛의 결과 값을 출력하기 전에 해당 픽셀의 Z 값이 Z축 클리핑 영역 안에 존재하는지 여부를 판단함으로써 스팬처리 유닛의 앞단에서 처리하지 못한 Z축 클리핑을 진행한다.

제안하는 구조를 따르면 클리핑 영역의 좌 우변 모두 걸쳐 있는 경우에 클리핑 영역 좌변의 좌측으로 벗어난 x의 길이만큼의 불필요한 연산을 진행하는 단점이 있지만 실질적으로 사용되는 3차원 물체를 렌더링 할 때 그와 같은 경우가 발생할 확률이 거의 없기 때문에 큰 문제가 되지 않는다. Z축 클리핑을 할 때 이미 삼각형의 많은 부분이 Z축 클리핑 영역을 벗어나 있는 경우에도 X축과 Y축 클리핑 영역을 벗어나 있지 않다면 앞에서 언급했던 불필요한 연산을 진행하는 경우가 발생하는 것은 실질적인 오버헤드라고 할 수 있다. 그러나 이 오버헤드에 비하여 제안하는 구조는 많은 연산 사이클과 연산기 수를 줄일 수 있는 이점이 있다.

III. 설계 및 검증

제안한 클리핑 알고리즘을 적용한 3차원 그래픽 엔진을 Verilog-HDL을 이용하여 설계하고 FPGA에 구현하였다. 표 1은 그림 3에서 언급한 기존의 알고리즘과 제안하는 알고리즘을 적용한 구조에 대한 설계 결과물을 Synopsys Design Compiler와 0.25um CMOS 표준 셀 라이브러리를 이용하여 100MHz 동작주파수를 목표로 합성한 결과와 연산 잠복기(latency)를 보여주고 있다. 기존 구조의 클리핑 엔진에서는 최소 16 사이클이 소요되고 재구성할 삼각형의 개수가 늘어난다면 연산 사이클은 최대 49 사이클까지 늘어나게 된다. 기존의 클리핑 엔진에 비해 제안하는 컬링 정렬 유닛을 사용할 경우 클리핑 엔진의 최소 사이클 16 사이클보다 15 사이클 감소된 1 사이클을 필요로 한다. 또한, 스캔 변환 엔진에서는 기존 삼각형 셋업 엔진에서 처리했던 삼각형 형성을 컬링 정렬 유닛에서 함께 처리하도록 설계함으로써 1 사이클을 감소시켰고 변처리 유닛과 스캔변환 유닛에서는 간단한 비교 연산만이 필요하기 때문에 사이클의 추가 없이 클리핑 연산을 진행할 수 있도록 하여 13 사이클에서 12 사이클로 한 사이클을 줄일 수 있다. 그러므로 전체 3차원 그래픽 엔진의 잠복기는 기존 구조의 최소 47 사이클에서 제안하는 구조의 31 사이클로 약 34% 감소됨을 확인할 수 있었다. 면적은 기존 구조에 비해 제안한 구조로 설계하였을 때 클리핑 엔진을 포함한 기하 변환 엔진에서는 89,100 게이트에서 63,300 게이트로 29% 감소하는 효과를 얻을 수 있었으며 실질적인 클리핑 기능을 수행하는 스캔 변환 엔진에서는 172,300 게이트에서 177,800 게이트로 약 3%가 증가했다. 그러므로 전체적으로는 261,300 게이트에서 241,000 게이트로 약 8% 정도의 면적이 감소된 것을 확인할 수 있다.

표 1. 기존의 구조와 제안한 구조의 구현 결과 비교
Table 1. Comparison of implementation results of the conventional and the proposed architecture.

	Conventional Architecture		Proposed Architecture	
	Area [gates]	Latency [cycle]	Area [gates]	Latency [cycle]
Transformation	56,900	18	56,900	18
Clipping	32,100	16~49	6,300	1
Triangle setup	90,800	10	90,700	9
Edge walk	69,500	2	71,500	2
Span processing	12,000	1	15,600	1
Total	261,300	47	241,000	31

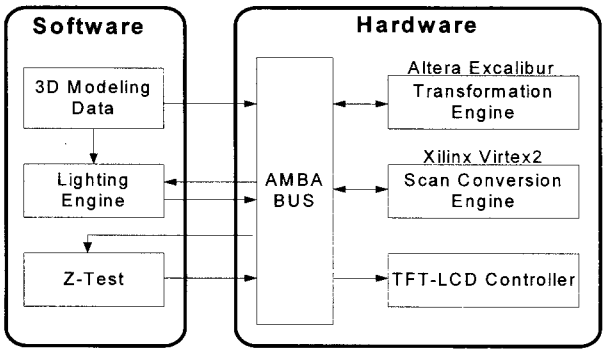

그림 7. 검증 시스템의 블록도
Fig. 7. Block diagram of Verification system.

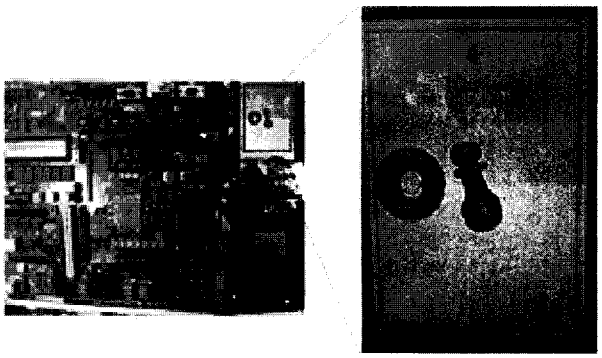

그림 8. 검증 시스템과 출력 영상
Fig. 8. The image of the verification system.

설계된 3차원 그래픽 엔진은 그림 7과 같이 검증 시스템을 구성하여 검증을 진행하였다. 입력 데이터는 각각 1,518, 504, 504, 1,020 개의 정점 정보를 가지고 있는 Pawn, Sphere, Cone, Torus의 총 4개의 모델을 사용하였다. 소프트웨어로부터 3차원 모델 데이터를 입력받고 기하변환 연산을 거쳐 다시 프로세서로 보내진 기하변환 결과 값은 소프트웨어로 처리된 조명 연산 결과 값과 함께 하드웨어로 구현된 스캔 변환 엔진으로 보내진다. 스캔 변환 연산을 마치면 픽셀 단위의 결과 값들이 다시 소프트웨어에서 Z-Test를 거친 후 최종 픽셀 데이터가 되어 TFT-LCD로 전달된다. 그림 8에 검증시스템과 TFT-LCD에 출력된 영상이 나타나 있다.

Ⅳ. 결 론

본 논문에서는 저전력 3차원 그래픽 엔진에 적합한 효율적인 클리핑 구조를 제안하고 설계 및 검증을 진행하였다. 일반적으로 기하연산 단계에서 많은 연산 사이클과 연산기를 요구하는 복잡한 클리핑 연산을 두 단계로 나누어 기하 변환 단계에서 삼각형 구성과 클리핑 영역을 완전히 벗어난 삼각형만을 제거하고 스캔 변환 단계에서 클리핑 영역에 걸쳐 있는 삼각형에 대한 처리를 함으로써, 기하변환 단계에서 연산 사이클과 연산기 수를 대폭 줄이고 스캔 변환 단계에서는 연산 사이클의 추가 없이 비교기와 컨트롤 레지스터의 추가만으로 처리할 수 있다. 이를 통해 전체적인 3차원 그래픽 프로세서의 면적과 사이클을 줄일 수 있었으며 파이프라인 스톨을 줄여 동작의 효율성을 증가시켰다. 제안하는 클리핑 구조를 적용한 3차원 그래픽 엔진의 기하 변환 엔진과 스캔 변환 엔진을 Verilog-HDL을 이용하여 설계하고 각각 FPGA를 사용하여 구현하였다. 합성 결과 기존 클리핑 엔진을 적용하였을 때에 비해 면적은 8%, 전체 잠복기는 34% 감소하였다.

참 고 문 헌

[1] Gopi K. Kolli, "*3D Graphics Optimizations for ARM Architecture*", Game Developers Conference, San Jose, CA, March 2004.
[2] 최윤철, 임순범, 고견, "*컴퓨터 그래픽스 배움터*", 생능출판사, p.27, 2003
[3] 이은곤, "*3차원 그래픽이 온라인 쇼핑몰 소비자의 가상현실 경험에 미치는 영향*", 연세대학교 석사학위 논문, p.152, 2003년 2월.
[4] Jaewan Bae, Donghyun Kim, Lee-Sup Kim, "*An 11M-triangles/sec 3D Graphics Clipping Engine for Triangle Primitives*", ISCAS, Kobe, May 2005.
[5] F. Devai, "*An analysis technique and an algorithm for line clipping*", IEEE Conference. on Information Visualization, London, pp.29-31, July 1998.
[6] Y-D. Liang, B. Barsky, "*A New Concept and Method for Line Clipping*", ACM Transactions on Graphics, Vol. 11, pp.276-290, January 1984.
[7] Tomas A. Moller, Eric Haines, "*Real Time Rendering*", A K Peters, p.30, p.95-96, 2002.
[8] 이지명, "*이동통신용 2D/3D 그래픽을 위한 기하 변환 엔진 설계*", 숭실대학교 석사학위 논문, 2007년 6월.

저 자 소 개

이 찬 호(정회원)
대한전자공학회 논문지
제43권 SD편 제9호 참조
현재 숭실대학교 정보통신전자공학부 교수