삼각형 셋업을 위한 파이프라인드 나눗셈 연산기 설계

정웅, 이문기, 한탁돈*
연세대학교 전기전자공학과, *연세대학교 컴퓨터공학과
전화 : 02- 2123-4731 / 핸드폰 : 019-554-5894

Design of the Pipelined Division Unit for Triangle Setup

W. Jeong, M.K. Lee, C. H. Jeong, T. D. Han
Department of Electrical & Electronic Engineering, Yonsei University
E-mail : woong@spark.yonsei.ac.kr

Abstract

This paper had been analyzed a triangle setup in 3-D graphics and the necessity of high throughput division unit.

And then this paper had been designed a pipelined division unit modifying series expansion algorithm and checked the error and performance of the designed division unit.

I. 서 론

3차원 그래픽은 멀티미디어 환경에서 점점 중요성이 증대되어지고 있는 응용분야이다. 3차원으로 모델링된 데이터를 하드웨어적으로 가속시키는 3차원 그래픽스 가속기는 크게 기하처리부(geometry processor)와 렌더러(renderer)로 이루어지며, 기하처리과정에서 처리하는 데이터 및 계산량의 막대함과 규칙성으로 인하여 범용 마이크로프로세서를 사용하기보다는 기하처리만을 위한 독립된 프로세서를 요구하게 된다.

범용 마이크로프로세서의 나눗셈 연산기는 다른 연산기에 비해 상대적으로 긴 대기시간(latency)을 갖도록 설계되지만, 기하처리부에서는 대기시간이 짧은 고성능의 나눗셈 연산기를 요구한다.

특히 트라이앵글 셋업 단계에서는 vertices 간의 색상과 좌표에 대한 증분을 계산하게 되는데, 이 단계에서는 나눗셈의 빈도수가 높아 기존 기하처리부를 사용하는 경우에는 병목현상이 발생하게 되므로 이 단계는 별도의 하드웨어로 구성되며, 높은 처리율을 가진 나눗셈기가 필요하다.

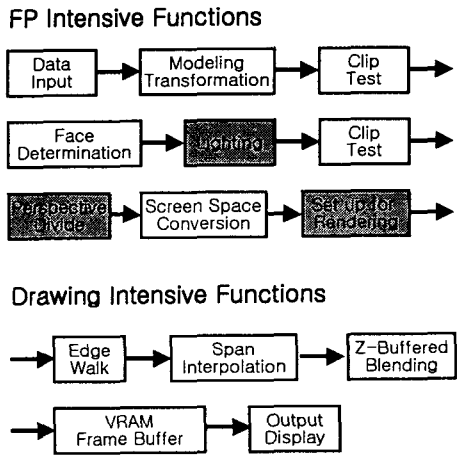
본 논문에서는 3차원 그래픽스 가속기의 기하처리과정과 트라이앵글 셋업단계를 살펴보고, 나눗셈 연산기의 성능 향상이 요구되는 단계를 분석하였다. 이를 바탕으로 트라이앵글 셋업 단계에 적합한 나눗셈 연산기를 급수 전개 알고리즘의 변형하여 설계하였다.

II. 트라이앵글 셋업 단계

2.1 3차원 그래픽 파이프라인

3차원 그래픽을 처리하는 파이프라인 단계는 아래 <그림1>과 같이 구성되어 있다. 3차원 그래픽 파이프라

인은 기하처리부와 렌더러 나눌 수 있으며, 기하처리부는 폴리곤 당 연산을 수행하는데, 부동소수점 연산을 수행하여 좌표를 변환하고, 시점을 바꾸며, 광원효과를 계산한다. 렌더러는 기하처리부에서 처리된 폴리곤 내의 픽셀의 색상값을 결정해주는 단계이다.



<그림 1> 표준 3차원 그래픽 파이프라인 [1]

좀 더 정교한 모델링을 위해서 한 장면 당 폴리곤의 개수는 점차 증대되고 있고, 이에 의해 막대한 대역폭과 연산처리량이 요구되어 기하처리부를 독립된 프로세서로 설계하는 것이 최근의 경향이다.

기하처리부는 보통 4개의 MAC (Multiply and Add) 유닛과 나눗셈/제곱근 연산기로 구성되어진다. 범용 마이크로프로세서의 나눗셈 연산은 그 빈도수가 적기 때문에, 나눗셈 연산기는 덧셈기나 곱셈기에 비해서 상대적으로 긴 대기시간(latency)을 갖도록 설계된다.

하나의 폴리곤 당 일정한 연산량을 가지며, 규칙적으로 연산이 반복되므로, 독립된 기하처리부는 3차원 그래픽 파이프라인에 적합하도록 최적화될 수 있다.

2.2 기하처리부 내의 나눗셈 연산 분석

기하처리부에 필요한 덧셈기와 곱셈기는 4개의 fMAC을 이용하는 것이 일반적이다. 이는 3차원 그래픽의 기본 좌표인 x, y, z, w 좌표로 이루어진 행렬 연산에 맞도록 설계된 것이다.

기하처리부의 나눗셈 연산기는 통상적으로 SRT radix-8의 나눗셈 연산기를 사용한다.

기하처리 단계에서 나눗셈 연산은 광원효과 처리 단계, 퍼스펙티브 변환, 트라이앵글 셋업 이렇게 세 곳에서 사용된다.

광원효과 처리 단계는 기하처리부에서 가장 긴 대기시간을 요구하는 단계이고, 많은 나눗셈 연산을 수행하며, 광원의 개수가 늘어날 때마다 급격하게 많은 계산량을 요구하지만, 각 광원에 대해서 나눗셈 뿐만 아니라 덧셈, 곱셈 등이 같은 비율로 증가하게 되기 때문에 나눗셈 연산기의 성능 향상에 의한 전체 성능 향상의 효과는 크지 않다. 이 단계에서의 나눗셈 연산기는 SRT divider가 적합하다.

퍼스펙티브 변환 단계에서는 각 폴리곤마다 한번의 나눗셈을 수행하게 된다. 따라서, 이 단계 역시 위의 SRT 나눗셈 연산기를 그대로 이용하게 된다.

삼각형 셋업 단계는 스캔 컨버젼(scan conversion)을 위한 파라미터 값을 계산하는 단계로 각 꼭지점 사이의 경사도(slope)와 RGBα 값의 증분값을 계산하는 단계이다. 이는 데이터의 특성상 별도의 파이프라인으로 구현된다.

2.3 트라이앵글 셋업 단계 분석

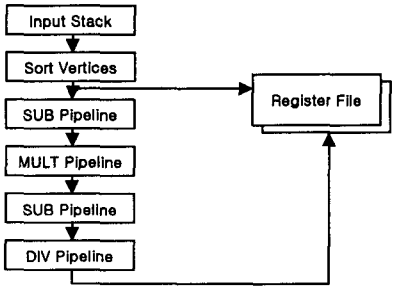
트라이앵글 셋업 단계는 스캔 컨버젼을 위한 셋업 파라미터를 계산하는 단계이다. 이 단계에서 계산하는 파라미터 값은 아래 <표1>과 같다.

Setup Parameters	Word 수
꼭지점 사이의 연결 정보	1 word
Edge interpolation increments: $\Delta x / \Delta y$ (edge 1, 2, 3)	3 words
색상, 깊이, 텍스쳐, 법선 Interpolation increments : $\partial r/\partial x, \partial g/\partial x, \partial b/\partial x, \partial \alpha/\partial x,$ $\partial r/\partial y, \partial g/\partial y, \partial b/\partial y, \partial \alpha/\partial y$	8 words

$\partial z/\partial x, \partial z/\partial y, \partial u/\partial x, \partial v/\partial y,$ $\nabla N(x), \nabla N(y)$	10 words
합계	22 words

<표2> Triangle Setup Parameter [1]

이 단계에서는 아래 그림과 같이 많은 나눗셈 연산을 연속적으로 처리하게 된다. 전체 파이프라인의 스톨 없이 이들 나눗셈 연산을 수행하려면, 파이프라인으로 이루어진 나눗셈 연산기(Pipelined divider)가 요구되며 대기시간보다는 처리율이 이 나눗셈기의 성능을 결정하는 중요한 요소가 되게 된다.



<그림 2> 트라이앵글 셋업 파이프라인

III. 제안된 나눗셈기의 구조

3.1 알고리즘

SRT 나눗셈 연산기는 파이프라인으로 이루어지기에 적합하지 않으며, 곱셈기를 이용한 나눗셈 연산기가 적합하다. 따라서, Newton-Raphson 나눗셈 연산기나 Taylor series expansion 알고리즘으로 구현되어진 나눗셈 연산기가 고려될 수 있다.

Newton-Raphson 알고리즘은 한 iteration 주기 내에 한번의 뺄셈과 두번의 곱셈을 수행한다. 이 연산들은 각각 dependent하기 때문에 parallel하게 처리할 수 없으며, pipeline으로 구현하는 경우에도 series expansion 알고리즘에 비해서 다양한 형태의 변형이 어렵다.

P. Hung은 Taylor series expansion 알고리즘을 다음과 같은 형태로 변형하였다. [2]

$$\frac{X}{Y} = \frac{X}{Y_h + Y_l} = \frac{X}{Y_h}\left(1 - \frac{Y_l}{Y_h} + \frac{Y_l^2}{Y_h^2} - \cdots\right)$$

$$\frac{X}{Y} \approx \frac{X\,(Y_h - Y_l)}{Y_h^{\,2}} \qquad \text{--- (1)}$$

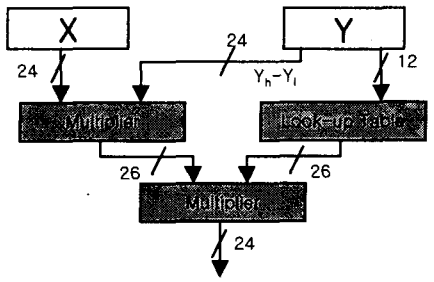
위 식에서 Y_h는 Y의 MSB부터의 상위비트이고, Y_l은 하위비트이다. 따라서, X에 (Y_h-Y_l)을 곱한후, $1/Y_h^2$을 곱하는 것으로 연산이 수행되어진다. 여기서 (Y_h-Y_l)은 <표3>와 같이 별도의 연산 없이 X와 (Y_h-Y_l)이 입력으로 들어가는 곱셈기의 Booth encoding의 변형으로써 가능하고, $1/Y_h^2$는 look-up table에서 읽어오게 된다. 위와 같이 Taylor series expansion algorithm을 변형하는 경우, 1차항과 2차항을 별도의 테이블에 저장하지 않고, 하나의 Look-up table에 기록하게 되어 look-up table의 크기를 줄일 수 있다.

3.2 나눗셈 연산기 구조

<표3> 변형된 (Y_h-Y_l)의 Booth 인코딩 [2]

Bits	Y_h Group	Y_l Group	Boundary		First Y_h	
			even	odd	even	Odd
000	0	0	0	0	0	0
001	+1	-1	-1	-1	0	+1
010	+1	-1	-1	+1	+1	+1
011	+2	-2	-2	0	+1	+2
100	-2	+2	+2	-2	-2	-2
101	-1	+1	+1	+1	-2	-1
110	-1	+1	+1	-1	-1	-1
111	0	0	0	-2	-1	0

위의 수식 (1)은 아래 <그림3>과 같이 구현이 가능하다.

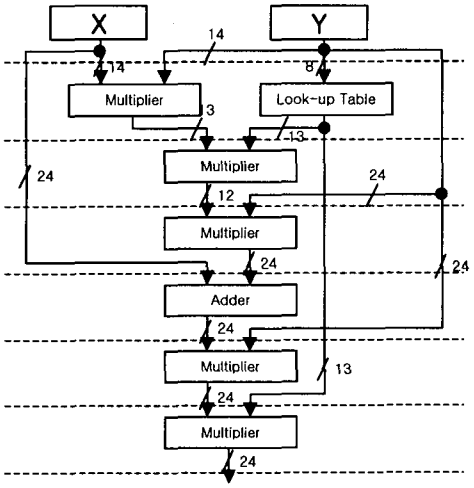


<그림3> P. Hung의 나눗셈 연산기 구조 [2]

위의 나눗셈 연산기는 single-precison인 경우에 한 번의 iteration과 2 싸이클만에 나눗셈을 수행할 수 있다. 하지만, LUT의 크기가 12.5KB를 요구하므로, 넓은 면적에 고통을 받게 된다. 트라이앵글 셋업 단계에서는 대기시간보다는 처리율이 우선하므로, <그림4>와 같이 변형할 수 있다.

위와 같은 변형을 통하여 다음과 같은 변화를 얻을 수 있다.

1) Look-up table의 크기를 12.5KB에서 544B로 줄일 수 있다.
2) 대기시간이 길어지고, 처리율은 동일하다.



<그림4> 제안하는 나눗셈 연산기 구조

IV. 결론

본 논문에서는 트라이앵글 셋업에 적합한 나눗셈 연산기의 구조에 대해서 연구하였다. 제안된 연산기는 동일한 크기의 LUT을 첨가하면, 제곱근 연산도 동일한 알고리즘을 통하여 수행할 수 있다. 곱셈기를 수정하고 약 1KB의 롬테이블을 추가하므로써, 대기시간 6싸이클/처리율 1싸이클의 나눗셈을 수행할 수 있다.

참고문헌

[1] A. Kugler, "The Setup for Triangle Rasterization", 11th Eurographics Workshop on Computer Graphics Hardware, August 26 1996

[2] P. Hung, H. Fahmy, O. Mencer, M.J. Flynn, "Fast division algorithm with a small lookup table", Signals, Systems, and Computers, 1999. Conference Record of the 33rd Asilomar Conference