논문 2009-04-01

3차원 그래픽의 트랜스포메이션을 위한 24-bit 부동 소수점 MAC 연산기의 설계

(A Design of 24-Bit Floating Point MAC Unit for Transformation of 3D Graphics)

이 정 우, 김 우 진, 김 기 철*
(Jungwoo Lee, Woojin Kim, Kichul Kim)

Abstract : This paper proposes a 24-bit floating point multiply and accumulate(MAC) unit that can be used in geometry transformation process in 3D graphics. The MAC unit is composed of floating point multiplier and floating point accumulator. When separate multiplier and accumulator are used, matrix calculation, used in the transformation process, can't use continuous accumulation values. In the proposed MAC unit the accumulator can get continuous input from the multiplier and the calculation time is reduced. The MAC unit uses about 4,300 gates and can be operated at 150 MHz frequency.

Keywords : 3D Graphics, 24-bit, Floating Point, MAC

Ⅰ. 서 론

휴대성과 편리성이 중요시되는 오늘날 여러 휴대정보기기가 개발되고, 어플리케이션의 내용도 다양해지고 있다. 그 중에서도 3차원 그래픽은 공학, 의학 등의 전문적인 분야뿐만 아니라 게임, 네비게이션 등 실생활에서 사용되고 있고, 그 활용 범위는 점점 확대되고 있다[1]. 현재 3차원 그래픽을 완전히 지원하는 대상은 일반 PC나 데스크탑이지만 점점 휴대정보기기로 확대되고 있는 추세이다.

대부분의 현대인들은 다양한 종류의 휴대정보기기를 보유하고 있으며, 이를 이용하여 많은 정보를 공유하고 여러 가지 컨텐츠를 제공받기를 원한다. 하지만 휴대정보기기의 연산능력은 데스크탑에 비하여 상대적으로 매우 제한적이기 때문에 많은 연산이 필요한 3차원 그래픽을 처리하기 위해서는 별도의 3차원 그래픽 가속기를 탑재해야 한다[2].

3차원 그래픽 가속기는 수많은 부동소수점 연산을 통하여 3차원 그래픽 처리 과정을 수행하며, 3차원 모델들의 위치정보와 색상정보를 구하게 된다.

3차원 그래픽 가속기에서의 처리 과정은 크게 모델의 정보를 계산하는 지오메트리 과정과 모델에 색상을 입히는 래스터라이져 과정으로 나뉘며, 지오메트리 과정은 다시 모델의 좌표를 계산하는 트랜스포메이션 과정과 모델의 색상을 결정하는 라이팅 과정으로 나뉘게 된다[4].

본 논문에서는 3차원 그래픽 처리 과정 중 트랜스포메이션 과정을 가속하기 위한 24-bit 부동소수점 MAC 연산기를 제안한다. 트랜스포메이션 과정은 주로 부동소수점 매트릭스 곱셈 연산으로 이루어져 있기 때문에 부동소수점 MAC 연산기를 이용하면 효과적으로 트랜스포메이션 과정을 수행할 수 있다.

부동소수점 MAC 연산기는 많이 존재하지만 3차원 그래픽을 위한 부동소수점 연산기는 매우 적은 편이며, 대부분 32-bit IEEE 754 단정도 형식을 가지고 있다[10]. 본 논문에서는 OpenGL의 유효자리를 만족시키며 32-bit 연산기보다 크기를 줄일 수 있는 24-bit 부동소수점 MAC 연산기를 제안한다[9]. 현재 OpenGL의 유효자리를 만족시키는

* 교신저자(Corresponding Author)
논문접수 : 2009. 01. 05., 수정일 : 2009. 02. 24., 채택확정 : 2009. 03. 16.
김기철 : 서울시립대 전자전기컴퓨터공학부 교수
※ 본 논문은 지식경제부 출연금으로 ETRI, SoC 산업진흥센터에서 수행한 IT-SoC 핵심설계인력 양성사업과 2008년도 「서울시 산학연 협력사업」의 「나노IP/SoC설계기술혁신사업단」, IDEC의 부분 지원으로 연구하였음.

그래픽용 24-bit 부동소수점 MAC 연산기는 알려져 있지 않다. 제안하는 MAC 연산기는 구조적으로 부동소수점 곱셈기 부분과 부동소수점 가산기 부분으로 나눌 수 있으며, 3차원 그래픽 매트릭스 연산에 최적화되어져 있기 때문에 효율적인 연산을 수행할 수 있다.

본 논문은 [15]에 발표된 연구내용을 확대 보완하여 작성되었다. 본 논문의 나머지 부분은 다음과 같이 구성되어 있다. II장 에서는 3D 그래픽 트랜스포메이션 과정에 대해 설명한다. III장에서는 24-bit 부동소수점 형식과 제안하는 MAC 연산기의 구조에 대해 설명한다. IV장에서는 트랜스포메이션 처리에 사용되는 matrix 연산과정에 대해 설명한다. V장에서는 제안한 구조의 성능에 대해 설명하고, VI장에서 본 논문의 결론을 맺는다.

II. 3D 그래픽 트랜스포메이션 과정

II장 에서는 3D 그래픽의 전반적인 처리 과정에 대해서 설명하고, MAC 연산기를 이용할 수 있는 트랜스포메이션 과정에 대해 상세히 설명한다.

1. 3차원 그래픽 처리 과정

3차원 그래픽 처리 과정은 크게 어플리케이션(application) 과정, 지오메트리(geometry) 과정, 래스터라이져(rasterizer) 과정으로 구성되어 있으며 그림 1과 같다[5].

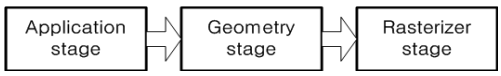

그림 1. 3차원 그래픽 처리 과정
Fig 1. 3D Graphic processing

그림 1에서 어플리케이션 단계는 그래픽 처리할 3차원 모델을 지오메트리 단계로 넘겨주는 역할을 주로 하며, 소프트웨어로 실행된다. 지오메트리 단계는 크게 정점의 위치와 방향을 결정해 주는 트랜스포메이션 과정과 정점의 색상을 계산하는 라이팅 과정으로 나뉘며, 모델의 정점들과 법선 벡터들을 변환하고 정점들의 색상을 구하는 단계이다[4]. 마지막으로 래스터라이져 단계에서는 모든 기하 요소들이 윈도우상의 픽셀들로 변환된다. 또한 텍스쳐(texture)와 연관된 기하 요소들은 해당 텍스쳐와 함께 랜더링(rendering) 된다.

2. 트랜스포메이션 과정

트랜스포메이션 과정은 모델/시야 변환(model/view transform)과, 투영(projection), 클리핑(clipping), 화면 매핑(screen mapping)으로 구성되며, 처리 순서는 그림 2에 나타나 있다[5][7].

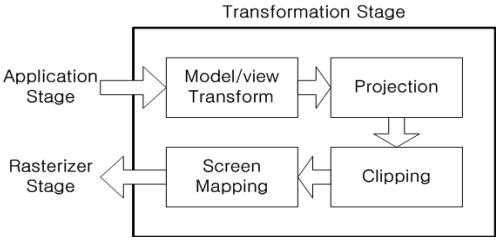

그림 2. 3차원 그래픽 지오메트리 과정
Fig 2. 3D Graphic geometry stage

모델 변환은 고유 공간의 3차원 모델을 하나의 월드 공간으로 옮기는 변환이며, 시야 변환은 모델들을 카메라 공간 또는 시야 공간으로 옮기는 변환이다[5]. 모델 변환과 시야 변환은 평행이동 행렬, 회전 행렬, 크기 조정 행렬, 쉬어 행렬 등으로 구성된다. 모델 변환과 시야 변환은 효율을 높이기 위해 일련의 행렬들을 하나의 행렬로 결합하고, 한 번의 매트릭스 곱셈을 통하여 연산을 수행한다. 식 (1)처럼 모델/시야 변환을 통하여 새로운 4×1 좌표 행렬을 얻게 되며, 4×4 모델 행렬의 변수 값은 이동, 회전, 크기 변환의 유형에 따라 변하게 된다.

$$\begin{pmatrix} ABCD \\ EFGH \\ IJKL \\ 0001 \end{pmatrix} \begin{pmatrix} X_{Old} \\ Y_{Old} \\ Z_{Old} \\ W_{Old} \end{pmatrix} = \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} \quad (1)$$

투영은 직교 투영 방법과 원근 투영 방법이 있으며, 모델/시야 변환과 마찬가지로 한 번의 매트릭스 곱셈을 통하여 연산을 수행한다. 식 (2)처럼 투영을 통하여 새로운 4×1 좌표 행렬을 얻게 되며, 4×4 모델 행렬의 변수 값은 투영의 종류에 따라 변하게 된다.

$$\begin{pmatrix} ABCD \\ EFGH \\ IJKL \\ MNOP \end{pmatrix} \begin{pmatrix} X_{Old} \\ Y_{Old} \\ Z_{Old} \\ W_{Old} \end{pmatrix} = \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} \quad (2)$$

클리핑은 시야 영역에서 벗어난 모델들을 제거하는 과정이다. 클리핑은 래스터라이져 과정에서 처리하는 것도 가능하며, 그것이 더 효율적이기 때문에 본 논문에서는 클리핑은 래스터라이져 과정에서 수행한다고 가정한다. 화면 매핑은 3차원의 데이터들을 래스터라이져 단계로 넘겨주기 위해 2차원의 윈도우 좌표계로 변환하는 과정이다. 식 (3)처럼 화면 매핑을 통하여 새로운 x, y 좌표를 얻게 되며, 2×3 모델 행렬의 변수 값은 화면의 크기에 따라 변하게 된다.

$$\begin{pmatrix} S_x & 0 & t_x \\ 0 & S_y & t_y \end{pmatrix} \begin{pmatrix} X_{Old} \\ Y_{Old} \\ 1 \end{pmatrix} = \begin{pmatrix} X \\ Y \end{pmatrix} \quad (3)$$

Ⅲ. 24-bit 부동소수점 MAC 연산기

III장에서는 24-bit 부동소수점 형식을 설명하고, 제안하는 MAC 연산기의 구조 및 동작에 대해 설명한다.

1. 24-bit 부동 소수점 형식

3D 그래픽 처리 과정 중 트랜스포메이션 과정은 모델의 위치를 계산하는 과정으로 연산 중 발생되는 오차로 인해 화질에 직접적인 영향이 미칠 수 있다. 3D 그래픽 표준인 OpenGL에서는 3차원 그래픽에서 허용하는 연산 오차 범위를 10-5 이내로 규정하고 있다[9]. 이러한 이유로 모델의 위치를 계산하는 트랜스포메이션 과정에는 표현범위가 넓고 세밀한 부동소수점 형식을 이용한다.

부동소수점 형식으로 많이 알려진 형식은 IEEE 754 단정도 형식이다[10]. IEEE 754 단정도 형식은 그림 3에 나타나 있으며, 부호부분(S) 1 bit, 지수부분(E) 8 bit, 가수부분(F) 23 bit로 구성되어진 32 bit의 데이터 형식이다.

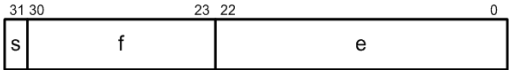

그림 3. IEEE 754 단정도 형식
Fig 3. IEEE 754 single-precision format

IEEE 754 단정도 형식은 hidden bit를 포함하며, 지수부분은 바이어스 처리된 형식으로 식 (4)와 같이 나타낼 수 있다.

$$F = (-1)^s \times 1.f \times 2^{e-127} \quad (4)$$

본 논문에서는 IEEE 754 단정도 형식을 토대로 한 24 bit 부동소수점 데이터 형식을 사용한다. 본 논문에서 사용한 24 bit 부동소수점 데이터 형식은 그림 4에 나타나 있으며, 부호부분(S) 1 bit, 지수부분(E) 7 bit, 가수부분(F) 16 bit로 구성된다.

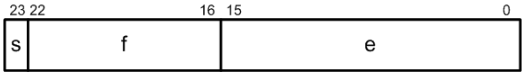

그림 4. 24-bit 부동 소수점 형식
Fig. 4. 24-bit floating point format

본 논문에서 사용한 24 bit 부동소수점 데이터 형식은 IEEE 754 단정도 형식과 마찬가지로 hidden bit를 포함하며, 지수부분은 바이어스 처리된 형식으로 식 (5)와 같이 나타낼 수 있다.

$$F = (-1)^s \times 1.f \times 2^{e-63} \quad (5)$$

2. MAC 연산기 구조

본 논문에서 제안하는 MAC 연산기의 블록도는 그림 5와 같다. 그림 5에서 나타난 것과 같이 제안한 MAC 연산기는 곱셈의 결과 값을 계속 누적하여 최종적인 결과 값을 출력하게 된다. 제안한 MAC 연산기는 매트릭스 곱셈시 발생되는 파이프라인 스톨(stall) 현상을 피하기 위한 두 가지 특징을 가지고 있다. 첫 번째는 매트릭스 연산의 순서를 조절하여 보다 효율적인 연산이 가능하도록 설계되었다. 두 번째는 지연 시간을 최소한으로 줄이기 위하여 3 stage로 구성되어 있다. 파이프라인 stage의 수가 많으면 동작주파수는 올라가지만, 매트릭스 곱셈시 곱셈 결과 값을 기존의 값에 누적시킬 때, 그 지연 시간으로 인해 스톨 현상이 발생되게 된다. 이를 피하고 동작주파수를 최대화하기 위하여 3 stage로 설계되었다. MAC 연산기의 각 stage는 부호, 지수, 가수 부분으로 나누어 계산되는데 첫 번째 stage에서는 곱셈의 전반부 과정이 진행되고 두 번째 stage에서는 곱셈의 후반부 과정과 덧셈의 전반부 과정이 동시에 진행되며, 세 번째 stage에서는 덧셈의 후반부 과정과 정규화 과정이 진행된다. 각 stage의 자세한 동작은 다음과 같다.

첫 번째 stage에서는 곱셈의 전반부 과정이 수

행된다. 먼저 두 입력 값을 받아서 부호를 결정해주고 덧셈기를 통하여 지수 값을 결정해주게 된다. 가수부분은 먼저 정규화 된 값을 실제 값으로 바꿔주는 비정규화 과정을 거친 후에 두 가수 부분을 곱해주게 된다. 그 후, 첫 번째 stage를 거친 값은 S_Reg, E_Reg, M_Reg에 저장이 된다.

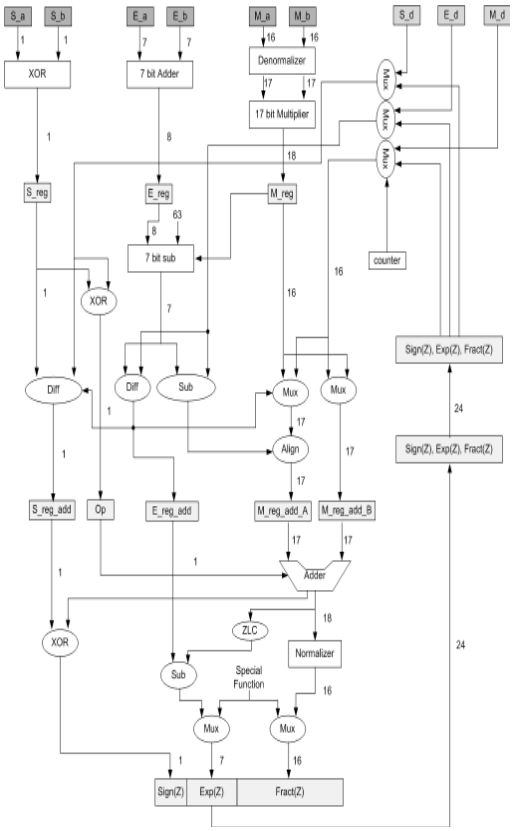

그림 5. MAC 연산기의 블록도
Fig. 5. A Block-diagram of MAC unit

두 번째 stage에서는 곱셈의 후반부 과정과 입력 값이나 가산기의 누적된 값 중 하나를 선택해서 덧셈의 전반부 과정이 수행된다. 첫 번째 stage를 거친 후 E_Reg에 저장되어져 있는 지수 값은 첫 번째 stage의 지수 덧셈으로 인해 바이어스 값이 2번 더해진 값이다 따라서, 두 번째 stage에서는 이중으로 바이어스 된 값을 올바르고 정정해주기 위해 E_Reg에 저장되어져 있는 지수 값에서 바이어스 값을 빼주게 된다. 이로써 곱셈의 후반부 과정이 끝나게 되고, 동시에 덧셈의 전반부 과정이 진행되게 된다. 부동소수점 덧셈을 위해서 먼저 두 값 중 지수 부분이 큰 값을 기준으로, 지수가 작은 수의 가수 부분을 두 수의 지수 차만큼 시프트 시켜서 유효자리수를 정렬하다. 한다. 이를 위해 지수 정렬 부분(Sub), 유효자리 가수 정렬 부분(Align)으로 구성되어 있으며, 부호 비트를 결정하기 위한 부호 부분(XOR, Diff)이 있다. 모든 연산이 끝나게 되면, 연산의 결과는 S_reg_add, E_reg_add, M_reg_add에 저장이 된다. 추가적으로 두 번째 stage에는 곱셈의 결과 값과의 덧셈을 하기위한 다른 하나의 입력 값으로 누적된 값을 이용할지 새롭게 들어오는 입력 값을 이용할지를 결정하기 위한 4 bit 카운터가 존재한다. 카운터의 경우 초기에는 0부터 시작을 하지만 4개의 결과 값이 나온 후에 카운터 값을 초기화 시킬 때 0이 아닌 2로 초기화 시킨다. 초기에는 곱셈의 결과 값을 가산기의 입력으로 사용해야 되기 때문에 2 cycle의 간격을 두어야 하고, 이 후에는 연속적으로 입력 값이 들어와서 계산이 되어져 가산기의 입력 값이 연속적으로 들어오기 때문에 카운터의 전체 count 수가 줄어들게 된다. 따라서 0이 아닌 2의 값으로 초기화 시켜준다.

마지막으로 세 번째 stage에서는 덧셈의 후반부 과정 및 정규화 과정이 진행된다. 세 번째 stage는 덧셈의 부호 부분(XOR), 지수 부분의 정규화 및 예외처리 부분(ZLC, Sub & Special Function), 가수부분 유효자리 가산(ADDER)과 정규화(Normalizer) 부분으로 구성되어져 있으며, 가수부분의 덧셈 및 정규화 과정을 수행하게 된다. 정규화 과정이 끝난 결과 값은 출력 값으로 나갈지 누적되기 위해 레지스터에 저장할지 결정하게 된다. 만일 값을 계속 누적시킨다면 결과 값은 레지스터에 저장되며, 이를 위해 24 bit 레지스터가 2개 사용 되었다.

제안한 MAC 연산기는 부동소수점 곱셈 과정과 덧셈과정에서 overflow와 underflow가 발생될 경우, 예외처리를 해주게 된다. 지수 값을 비교해서 overflow가 발생되면 7 bit에 해당하는 지수 값은 “1111110”으로, 가수 값은 “1111111111111111”로 고정한다. underflow의 경우에는 지수 값은 “0000000”으로, 가수 값은 “0000000000000000”으로 고정한다[11]. 즉, overflow가 발생되면 최대값으로 정정되며, underflow가 발생되면 최소값으로 정정되게 된다.

제안하는 MAC 연산기에 사용되는 라운딩 방법은 곱셈과정에서는 버림을 사용하고 덧셈과정에서는 유효자리 부분 연산의 유효숫자 이하의 자릿수의 각 bit를 OR 연산을 통해 '1'이 되면 올림을 하는 sticky 방법을 사용한다.

Ⅳ. MAC 연산기를 이용한 트랜스포메이션

본 논문에서는 제안한 MAC 연산기를 이용하여 3차원 그래픽 트랜스포메이션 과정을 수행한다. 트랜스포메이션 과정은 식 (1), (2), (3)처럼 주로 매트릭스 곱셈 연산으로 구성되어져 있기 때문에 제안한 MAC 연산기를 이용하면 효율적인 연산이 가능하다. 식 (6)은 일반적인 매트릭스 곱셈 과정을 나타낸 식이다.

$$\begin{pmatrix} ABCD \\ EFGH \\ IJKL \\ MNOP \end{pmatrix}\begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix}=\begin{pmatrix} AX+BY+CZ+DW \\ EX+FY+GZ+HW \\ IX+JY+KZ+LW \\ MX+NY+OZ+PW \end{pmatrix} \quad (6)$$

3차원 그래픽 처리 과정에서 입력으로 들어오는 정점 좌표(x, y, z, w)는 사원수로 구성되기 때문에 w의 값은 항상 1이다. 따라서 식 (6)은 식 (7)처럼 나타낼 수 있다.

$$\begin{pmatrix} ABCD \\ EFGH \\ IJKL \\ MNOP \end{pmatrix}\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}=\begin{pmatrix} AX+BY+CZ+D \\ EX+FY+GZ+H \\ IX+JY+KZ+L \\ MX+NY+OZ+P \end{pmatrix} \quad (7)$$

식 (7)은 3차원 그래픽 처리 과정 중 트랜스포메이션에 사용되는 기본적인 매트릭스 연산이다. W에 해당하는 값이 1로 되어 있기 때문에 3D 그래픽을 위한 연산에 특화되어 있고, 가산기와 곱셈기가 따로 되어 있는 구조에서의 매트릭스 연산 과정보다 훨씬 간단하게 연산의 결과 값을 얻을 수 있다. 위의 matrix 연산을 살펴보면 X와 곱해지는 값은 A, E, I, M으로 4×4 행렬의 첫 번째 열이고, 마찬가지로 Y, Z, W는 각각 두 번째, 세 번째, 네 번째 열과 곱해진다. 또한, 곱해진 값들은 각 각 더해져서 누적된 값을 최종 값으로 이용한다. 구조적으로 살펴보면 결과 값의 첫 번째 값을 얻기 위해 우선, 곱셈기를 이용하여 입력 값 A, X를 받아서 AX 값을 생성하고 그 값과 W와 곱해지는 입력 값 D를 가산기의 입력으로 받아서 AX+D값을 생성한다. 가산기의 결과 값과 입력 B, Y를 곱한 값을 가산기의 입력으로 받는다. 이러한 과정을 거쳐서 최종적으로 우리가 원하는 AX+BY+CZ+D의 matrix 연산의 결과 값을 얻을 수 있다.

본 논문에서는 matrix 연산의 결과 값을 얻기 위해 일반적으로 생각하는 AX+D 연산 후에 BY 값을 계산해서 AX+D 값에 누적시키는 것이 아니라 X에 곱해지는 A, E, I, M값을 먼저 계산한다. 계산되어진 값은 파이프라인 방식을 거쳐 연산이 끝나면 레지스터에 저장해 그 다음 연산과 누적 시킬 수 있는 방식을 이용한다.

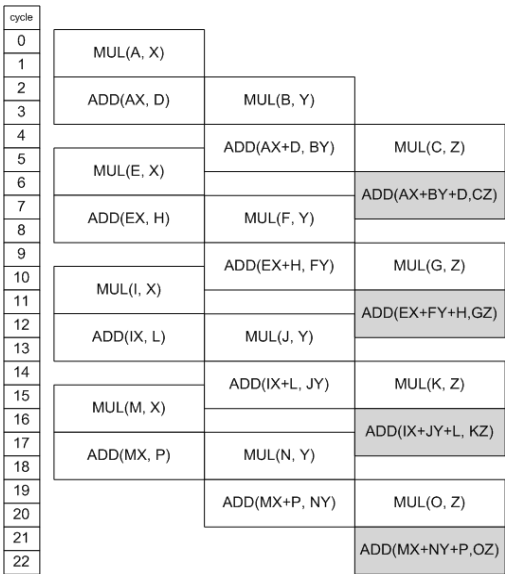

그림 6. 덧셈기와 곱셈기가 따로 되어 있는 구조에서의 MAC 연산 과정

Fig. 6. Calculation processing of general MAC

곱셈기와 가산기를 따로 사용하는 가산기와 곱셈기가 따로 되어 있는 구조에서의 매트릭스 연산에서는 그림 6과 같이 연산의 결과 값을 얻는데 22 cycle이 걸리고, 결과 값이 연속으로 나오지 않는다. 그림의 빈 칸은 입력 값이 연속적으로 들어오지 못하기 때문에 연산 도중 연산기가 다음 입력 값을 기다리는 부분이다. 따라서 연산에 걸리는 시간이 길어지고 중간에 연산을 쉬어야 하기 때문에 비효율적이다.

반면, 제안한 MAC 연산기 구조에서는 그림 7과 같이 결과 값 4개가 나오는데 총 14 cycle이 소요되고 12 cycle 마다 4개의 결과가 연속적으로 나온다. 또한, 입력 값을 연속으로 받아서 처리하기 때문에 연속적인 matrix 연산이 가능해 진다. 제안한 MAC 연산기를 이용하면 가산기와 곱셈기가 따로 되어 있는 구조에서의 MAC 연산기를 이용 했을 때보다 matrix 연산에 걸리는 시간을 줄일 수 있다.

cycle	INPUT	MUL1	MUL2/ADD1	ADD2
0	A, X, D			
1	E, X, H	A, X, D		
2	I, X, L	E, X, H	AX, D	
3	M, X, P	I, X, L	EX, H	AX+D
4	B, Y	M, X, P	IX, L	EX+H
5	F, Y	B, Y, AX+D	MX, P	IX+L
6	J, Y	F, Y, EX+H	BY, AX+D	MX+P
7	N, Y	J, Y, IX+L	FY, EX+H	AX+BY+D
8	C, Z	N, Y, MX+P	JY, IX+L	EX+FY+H
9	G, Z	C, Z, AX+BY+D	NY, MX+P	IX+JY+L
10	K, Z	G, Z, EX+FY+H	CZ, AX+BY+D	MX+NY+P
11	O, Z	K, Z, IX+JY+L	GZ, EX+FY+H	AX+BY+CZ+D
12		O, Z, MX+NY+P	KZ, IX+JY+L	EX+FY+GZ+H
13			OZ, MX+NY+P	IX+JY+KZ+L
14				MX+NY+OZ+P

그림 7. 제안한 MAC 연산기를 이용한 연산 과정
Fig. 7. Calculation processing of proposed MAC

Ⅴ. 실험 및 결과

제안한 MAC 연산기는 곱셈기와 덧셈기를 합쳐서 3 stage로 구성되었다. 곱셈기와 덧셈기가 따로 되어 있는 구조에서는 구조적으로 덧셈기 부분이 2 stage를 거쳐야 연산값이 나온다. 따라서 덧셈기의 입력으로 누적된 값과 입력 값을 동시에 받기 위해 입력 값을 연속으로 받지 못한다는 단점이 있다. 즉, 덧셈 부분의 연산을 수행하기 위해서 덧셈기의 결과 값을 기다려야 하는 상황이 발생한다. 이러한 단점을 해결하기 위해 제안한 MAC 연산기는 연산의 순서를 바꿔서 입력 값을 연속으로 받도록 하였다. 또한, 덧셈기의 결과 값을 레지스터에 저장하는 방법을 이용해서 덧셈기의 입력 값 두 개를 동시에 받도록 하였다.

표 1은 제안한 MAC 연산기와 덧셈기/곱셈기에 대한 Xilinx 합성 결과를 나타내고 있으며, 타겟 디바이스로 xc4vlx200-ff1513을 사용하였다.

표 2는 곱셈기와 덧셈기가 따로 되어 있는 구조와 제안한 MAC 구조에서의 성능 및 gate count를 비교한 표이다. gate count는 synopsys사의 design compiler를 이용하여 측정하였다. 표 1과 표 2에서 알 수 있듯이 제안한 MAC 연산기는 곱셈기와 덧셈기가 따로 되어 있는 구조와 비슷한 크기를 가지며 보다 좋은 성능을 보인다.

표 1. Xilinx를 이용한 검증
Table 1. Verification using xilinx

	Proposed MAC	Add+ Mul
Logic Utilization		
Number of Slice Flip Flops	138	58
Number of 4 input LUTs	589	561
Logic Distribution		
Number of occupied Slices	330	306
Number of Slices containing only related logic	330	306
Number of Slices containing unrelated logic	0	0
Total Number 4 input LUTs	706	585
Number of BUFG/BUFGCRLs	2	1
Total equivalent gate count for design	6,970	5,534
Additional JTAG gate count for IOBs	5,856	7,008

표 2. 성능 비교
Table 2. Performance comparison

	Proposed MAC	Add + Mul
1 Matrix (cycle)	12	20
1 Transformation (cycle)	30	47
1 초에 수행할 수 있는 Transformation (개/sec, 150MHz)	5,000,000	3,191,500
Frequency	150 MHz	150 MHz
Gate count	4,689	4,646

표 3. 타 연산기와의 비교
Table 3. Comparison to other MAC units

	MAC1 [12]	MAC2 [13]	MAC3 [14]	Proposed MAC
Format	IEEE 754 32-bit	IEEE 754 32-bit	IEEE 754 32-bit	24-bit
Operation	mul-acc	mul-acc, mul, mul-sub	mul-acc	mul-acc
Pipeline stage	1	1	2	3
Frequency	3 GHz	80 MHz	30 MHz	150 MHz
Gate count	230,000	12,800	56,000	4,689

표 3은 제안한 MAC 연산기와 다른 연산기와의 비교를 나타낸 표이다. 다른 MAC 연산기에 비하여 작은 크기를 가지며, 높은 주파수에서 동작하는 것을 확인할 수 있다.

Ⅵ. 결 론

본 논문에서는 3차원 그래픽 처리 과정 중 트랜스포메이션에 사용되는 24 bit 부동소수점 MAC 연산기를 제안하였다. 제안한 MAC 연산기는 곱셈기와 덧셈기를 합쳐 누산된 값을 연속적으로 이용할 수 있는 구조를 가지고 있다. 덧셈기와 곱셈기가 따로 되어 있는 구조에서의 matrix 연산의 경우에는 덧셈기와 곱셈기가 분리 되어 있다. 따라서 matrix 연산을 수행 시 연산에 필요한 입력 값을 기다리기 위해 연산기가 쉬는 경우가 발생한다. 반면 제안한 MAC 연산기에서는 곱셈기와 덧셈기가 따로 되어 있는 구조의 matrix 연산 수행 순서를 바꾸고, 레지스터를 이용한다. 하드웨어가 조금 추가되지만 곱셈기와 덧셈기를 따로 사용했을 때와 비교해 연산에 소요되는 시간을 줄이는 효과를 얻고 연산기가 쉬지 않고 연속적으로 연산이 가능하기 때문에 효율적이다. 제안한 MAC 연산기는 약 150MHz에서 동작가능하며, 4,700 gate의 크기를 갖는다.

참 고 문 헌

[1] 이민영, 이상호, 장선호, "3D 그래픽칩 기술 및 산업동향", System Semiconductor Forum, May, 2006.

[2] David H. Eberly, "3D Game Engine Design", Morgan Kaufmann, May, 2001.

[3] L. Garber, "The wild world of 3D graphics chips", IEEE Computer, vol. 33, no. 9, pp. 12-16, Sep. 2000.

[4] 주우석, OpenGL로 배우는 컴퓨터 그래픽스, 한빛미디어(주), 서울, 2006.

[5] Tomas Akenine-Moller, Eric Haines, Real-Time Rendering - second edition, A K Peters, Massachusetts, 2002.

[6] Samuel R.Buss, 3-D Computer Graphics - A Mathematical Introduction with OpenGL, Cambridge University press, United Kingdom, 2003.

[7] 남기훈, 하진석, 곽재창, 이광엽, "내장형 3D 그래픽 가속을 위한 부동 소수점 Geometry 프로세서 설계", 전자공학회 논문지, 제 43 권 SD 편 제 2호, pp. 108~117, 2006.

[8] Alan Watt, 3D Computer Graphics - Third edition, Pearson Addison Wesley, Edinburgh, 2000.

[9] http://www.opengl.org

[10] "IEEE Standard for Binary Floating-Point Arithmetic", ANSI/IEEE Standard 754-1985, New Yock, IEEE, Aug, 1985.

[11] Israel Koren, Computer Arithmetic Algorithms, pp.67~86, 2002.

[12] Sriram R. Vangal, Yatin V. Hoskote, Nitin Y. Borkar, Atila Alvandpour, "A 6.2-GFlops Floating-Point Multiply-Accumulator With Conditional Normalization", IEEE Journal of solid-state circuits, VOL. 41, NO. 10, Oct. 2006.

[13] "Floating Point Multiply Accumulate Unit", Silicon Logic Engineering.

[14] Daniel A. Staver, Chung-Yih Ho, Karl J. Molnar, Richard D. Baertsch, "A 30-MF LOPS CMOS Single Precision Floating Point Multiply/Accumulate Chip", ISSCC87, Feb. 1987.

[15] 김우진, 이정우, 김기철, "3차원 그래픽의 트랜스포메이션을 위한 24-bit 부동 소수점 MAC 연산기의 설계", 대한임베디드공학회 추계학술대회, pp.147~150, Nov. 2008.

저 자 소 개

이 정 우(Jungwoo Lee)

2005년 서울시립대 전자전기컴퓨터공학부 학사.
2007년 서울시립대 전자전기컴퓨터공학부 석사.
현재, 서울시립대 전자전기컴퓨터공학부 박사과정.

관심분야: 3D-Graphics Rendering, SoC 디자인, 멀티미디어 시스템.

Email: jjung@uos.ac.kr

김 우 진(Woojin Kim)

2006년 서울시립대 전자전기컴퓨터학부 학사.

현재, 서울시립대학교 전자전기컴퓨터학부 석사과정.

관심분야: 3D-Graphics Rendering, 병렬 프로세서, 멀티미디어 시스템.
Email: kimwj11@uos.ac.kr

김 기 철(Kichul Kim)

1982년 서울대 전기과 학사. 1984년 서울대 전기과 석사.
1991년 University of Southern California, Electrical Engineering System 박사.
현재, 서울시립대 전자 전기 컴퓨터 공학부 교수.
관심분야: 임베디드 시스템, 컴퓨터 시스템, 멀티미디어 시스템, 디지털 VLSI 디자인, 병렬 프로세싱.
Email: kkim@uos.ac.kr