Newton Method을 이용한 저비용 5-stage 멱승기의 구현

송세현[O] 김기철
서울시립대학교 전자전기컴퓨터공학부
wisdomofodin@gmail.com[O], kkim@uos.ac.kr

An Implementation of Low Cost 5-stage Powering Unit Using Newton Method

Sehyun Song[O] Kichul Kim
School of Electrical & Computer Engineering, University of Seoul

요 약

본 논문에서는 모바일용 3차원 그래픽 라이팅 엔진을 위한 부동소수점 멱승기를 제안한다. 3D 그래픽의 라이팅 과정은 연산량이 많고, 복잡하기 때문에 각 연산 유닛들이 저비용으로 빠르게 연산을 수행해야 한다. 본 논문에서 제안한 멱승기는 처리율을 높이기 위해 파이프라인 구조를 사용하였으며, 10^{-4} 의 정확도를 만족한다. 전체 구조는 5 stage로 구성되며, 크게 로그연산기와 지수연산기로 이루어져 있다. 일반적으로 로그연산기는 정확도를 높이기 위하여 큰 룸 테이블을 사용하는데, 이는 많은 면적을 차지하게 된다. 이러한 룸 테이블 면적 문제를 해결하기 위하여 Newton method을 사용하여 룸 테이블의 사이즈를 줄였다. 또한 오일러 상수를 밑으로 하는 지수연산기도 입력 비트의 크기를 줄이고, 테이블의 개수를 늘림으로써 룸 테이블의 크기를 줄였다. 지수연산의 밑은 부동소수점 포맷으로 [0,1]의 범위를 가지며, 승은 정수 포맷으로 [0,128]의 범위를 갖는다. Magnachip 0.18μm 공정에서 100Mhz의 동작주파수를 만족하였으며, 약 16k gates을 차지한다.

1. 서 론

최근에는 PDA(Personal Digital Assistant), 휴대폰 등 휴대 정보기기 및 휴대용 게임기의 수요가 급증하고 있다. 이러한 기기의 발전에 따라 휴대용 기기에서의 3차원 그래픽 처리 능력이 중요해 지고 있다. 3차원 그래픽 영상을 표현하기 위해서는 많은 데이터를 연산해야 하며 그 연산 과정 또한 복잡하다. 이를 범용 CPU 만으로 처리하기에는 연산 능력이 부족하기 때문에 그래픽 처리를 위한 가속 하드웨어를 별도로 설계 해야 한다. 특히 고감도 3차원 영상을 실현하기 위해서는 독립적인 지오메트리 처리 프로세서를 두는 것이 필수적이다.

지오메트리 엔진은 크게 트랜스포메이션 엔진과 라이팅 엔진으로 이뤄진다. 이 중 라이팅 엔진은 많은 연산량을 필요로 하며, 난반사 성분, 정반사 성분 그리고 주변광 성분에 의해 값이 결정된다[1]. 특히 정반사는 물체에 하이라이트를 생성함으로써 물체의 표면이 반짝거리도록 보이게 하는 역할을 하고, 이러한 하이라이트를 통하여 물체 표면의 굴곡을 표현하며, 광원의 방향과 위치를 결정 할 수 있도록 도와준다. 정반사에 대한 연산은 지수연산을 필요로 하는데, 밑의 값으로 물체를 통해 관찰자로 들어오는 빛의 강도를 가지며, 승으로 광택도를 가지게 된다. 또한 라이팅 엔진은 광원이 집중 조명형태로 존재할 때, 지수연산을 사용한다. 광원으로부터 수직에 있는 면이 더욱 강한 빛을 받게 되는데, 이러한 집중 조명광에서의 감쇠 정도를 나타내기 위하여 지수연산이 수행된다. 지수연산은 전체 연산량에 비해 연산 횟수가 많은 것은 아니지만, 연산 자체가 매우 복잡하기 때문에 연산 시간이 오래 걸린다. 본 논문에서 제안한 지수연산(A^B)은 A의 값으로 부동소수점 포맷을 가지며, 범위는 [0, 1]을 나타낸다. B는 정수 포맷을 가지며, 범위로 [0, 128]를 나타낸다.

본 논문의 구성은 다음과 같다. 본 절의 서론에 이어 제 2 절에서는 지수연산에 사용되는 알고리즘을 소개한다. 지수연산은 크게 로그연산과 오일러 상수(Euler's constant)를 밑으로 하는 지수 연산으로 이뤄지며, 로그연산은 뉴튼 방식(Newton method)을 이용하여 값을 구하게 된다. 제 3 절에서는 전체 구조에 대해서 설명하며, 각 주요 유닛별로 자세하게 설명한다. 마지막으로 제 4절에서 본 논문의 결론을 보인다.

2. 알고리즘

이번 절에서는 지수연산(A^B)에서 사용되는 알고리즘에 대해서 설명한다. 멱승기는 24비트 부동소수점 포맷을 가지는 A와 정수 포맷을 가지는 B을 입력으로 받는다. 두 입력값의 관계를 아래 식 (1)같이 나타낼 수 있다.

$$A^B = e^{B\ln(A)} \quad (1)$$

입력값 A는 부동소수점이므로, $A = M_A 2^{E_A}$ 로 나타낼 수 있다. 이때, M_A는 부동소수점 A의 가수부(Mantissa)를 나타내며, E_A는 지수부(Exponent)를 나타낸다. 이를 정리하면 다음 식 (2)와 같다.

$$\begin{aligned} A^B &= e^{B\ln(A)} \\ &= e^{B\ln(M_A 2^{E_A})} \\ &= e^{B(\ln(M_A) + \ln(2^{E_A}))} \\ &= e^{B(\ln(M_A) + E_A \ln(2))} \end{aligned} \quad (2)$$

식 (2)는 A와 B의 지수연산을 로그연산, 곱셈 그리고 오일러 상수(Euler's constant)를 밑으로 하는 지수연산으로 나타낼 수 있다. 이는 지수연산 자체의 어려움뿐만 아니라 B의 범위가 넓으므로 생기는 연산의 복잡함을 좀더 쉽게 연산할 수 있다는 장점이 있다. 설명을 좀더 쉽게 하기 위하여 새로운 기호를 사용한다. A의 가수부인 M_A는 오일러 상수를 밑으로 하는 로그연산을 수행하여 중간 값인 L을 생성한다. 또한 A의 지수 부분인 E_A는 ln(2)인 상수와 곱셈 연산을 수행하며, 결과로 M을 생성한다. L과 M은 덧셈연산을 수행하며, 그 결과로 입력값 B와 곱셈연산을 수행할 결과값 X을 생성한다. B와 X의 곱셈 연산의 결과는 오일러 상수를 밑으로 하는 지수연산의 입력값으로 사용되며, 그 결과 최종적인 결과값 P가 생성된다. 결과값 P는 고정소수점 포맷을 이루고 있으며, 이를 부동소수점 포맷으로 바꿔주기 위한 추가적인 유닛이 필요하다. 이는 3D 그래픽 연산에 사용되는 연산기들이 부동소수점 형태를 이루고 있기 때문에 호환성을 높이기 위함이다. 전체 유닛 설계 시 입력값 A는 24비트 부동소수점 포맷을 사용하며, B는 7비트 정수를 포맷으로 사용한다. 모바일 기기에서는 자원을 절약할수록 효과적인데, 일반적으로 사용되는 단정도(single precision)보다 24비트 포맷을 사용하여 자원을 절약 할 수 있으며, 정확도 역시 만족할 수 있다. [2].

2.1 로그연산 알고리즘

로그연산을 구현하는 방법으로는 룸 테이블을 이용하는 방법, 반복 수행 방법(iteration method), 구분적 선형 근사법(piecewise linear approximation method), 뉴튼 방법(Newton method)등이 있다. 룸 테이블을 이용하는 방법은 입력값에 대한 로그연산 값들을 메모리에 넣어두고, 추가적인 수행 시간 없이 바로 결과값을 알 수 있는 방법이다[3]. 이는 무척 빠른 연산을 수행한다는 장점이 있지만 자원을 많이 사용한다는 단점이 있다. 이에 반해 구분적 선형 근사법은 룸 테이블 방식보다는 저비용이다. 이 방법은 여러 개의 구획(segment)로 범위를 나누어 원하는 정확도 까지 값을 구해 가지만, 높은 정확도를 구하기가 힘들다는 단점이 있다[4]. 이뿐만 아니라 반복 수행 방법을 통하여 로그 연산값을 구할 수도 있다. 동일한 연산 과정을 반복해 가면서 더욱더 원하는 값에 근사해 가는 방식을 말하는데, 이때 초기 근사값이 중요하다. 만일 초기의 근사값이 원하는 결과값보다 많이 차이가 나게 되면, 여러 번의 반복수행(iteration)을 해야 하기 때문이다. 반복수행이 많아 질수록 결과값의 지연(latency)은 증가하게 된다. 이러한 반복 수행을 적게 하면서, 최대한 정확도를 높이기 위하여 룸 테이블 방식과 반복수행 방법의 하나인 뉴튼 방식을 혼합하여 사용하였다[5]. 이 방법은 룸 테이블을 이용하여 최종 결과값에 최대한 유사한 부분부터 연산을 시작하며, 뉴튼 방식을 이용하여 반복 수행을 하여 원하는 정확도까지 접근해 간다. 이러한 방법은 반복수행의 횟수를 줄일 수 있으며, 짧은 지연시간 안에 높은 정확도를 만족할 수 있다. 뉴튼 방식은 미분을 이용한 함수 $f(x)$의 값을 0로 만드는 방법이다. 몇 번의 반복을 통하여 $f(x)$의 값을 0으로 만들며, 이때 x값을 최종 결과 값으로 출력한다. 반복할수록 정확도는 올라가며, 원하는 정확도만큼의 반복을 수행한다. 아래 식 (3)은 뉴튼방식을 나타낸 것이다.

$$x_{i+1} = x_i + f(x_i)/f'(x_i) \quad (3)$$

위의 식 (3)에서 $f(x) = e^x - D$ 와 같이 주워질 때, $x = \ln(D)$에서 $f(x)$가 0의 값을 갖게 된다. 즉 식 (3)에 아래와 같은 식 (4)을 대입한다.

$$\begin{aligned} f(x_i) &= e^{x_i} - D \\ f'(x_i) &= e^{x_i} \end{aligned} \quad (4)$$

최종적인 뉴튼 공식은 아래 식 (5)와 같다.

$$x_{i+1} = x_i + De^{-x_i} - 1 \quad (5)$$

식 (5)에서 초기값인 x_0가 최종 결과값에 근사할수록 반복횟수를 줄일 수 있으며, 원하는 정확도까지 빠르게 접근할 수 있다. 그래서 룸 테이블을 이용하여 x_0을 구해준다. 입력값 D에 대해 9비트의 정확도를 가지는 로그연산을 수행하며, 그 결과로 x_0을 구하게 된다. 이를 식 (5)에 대입하여 한번의 반복 연산을 수행하고 나면, 최소 17비트의 정확도를 갖는 x_1을 구할 수 있다[5].

2.2 오일러상수 지수연산

2절에서 설명하였듯이 멱승기는 로그연산을 통해 구해진 X에 대하여 오일러상수를 밑으로 하는 지수연산을 수행한다. 밑의 값이 고정되어 있으므로, 일반 지수연산(A^B)보다 훨씬 간단하다. 일반적으로 오일러상수 지수연산은 룸 테이블을 이용한 테이블 유도 방식을 이용 할 경우 지연시간 없는 빠른 연산을

보여준다. 하지만 입력값의 다양함과 높은 정확도에 대한 요구 때문에 테이블의 크기가 매우 커지게 된다. 테이블을 줄이기 위하여 식 (6)과 같은 방법을 사용한다.

$$e^{x} = e^{(x_1+x_2)} = e^{x_1}e^{x_2} \qquad (6)$$

식 (6)은 입력값 x를 x_1과 x_2로 나누며, 이를 각각의 지수연산의 룸 테이블을 이용하여 구해준다. 입력의 비트수를 줄임으로써 전체 테이블의 크기를 줄 일수 있다. 하지만 룸 테이블 연산 이후, 두 결과값을 서로 곱해주어야 하므로 추가 지연시간이 생기게 된다.

3. 구 현

본 절에서는 2 절에서 설명한 알고리즘을 구현한다. 입력 A는 24비트 부동 소수점 포맷으로 구성된다. 입력 B는 7비트 정수 포맷이며, [0, 128]의 범위를 갖는다. 구조는 아래 그림 1과 같다. 전체 구조는 5 단계(stage)로 이루어져있고, 크게 로그연산기와 오일러 상수를 밑으로 하는 지수연산기로 구성된다. 입력 A의 가수부분(M_A)는 로그 연산기의 입력으로 사용되며, 2 cycles 이후에 결과값 L을 생성한다. A의 지수부분(E_A)은 상수값인 ln(2)와 곱해진다. 부동소수점 A의 범위가 [0, 1]이므로, 지수부분에서 입력이 1보다 크게 들어올 경우, 1로 수렴시킨다. 로그연산의 결과값 L과 지수부분의 연산 결과값 M을 더하여 X을 생성한다. 덧셈기의 입력 M은 5.18비트의 포맷으로 구성되며, 이는 정수 5비트와 소수 18비트의 고정소수점을 나타낸다. X는 멱승기의 입력값 B와 곱해지며, 오일러 상수를 밑으로 하는 지수연기의 입력으로 사용된다. 곱셈기의 결과 중 정수 4비트와 소수 14비트가 지수연산의 입력으로 사용된다. 곱셈기의 결과가 16 보다 클 경우 원하는 정확도 내에서는 0으로 수렴하기 때문에 이때에는 지수연산을 수행하지 않고 멱승기를 통하여 최종 결과값이 0로 수렴하게 하였다. 최종 결과의 부호는 입력값 A의 부호와 B의 가장 낮은 비트(LSB)를 논리곱(and)을 통하여 결정한다. 홀수 승일경우 A의 부호를 유지하고, 짝수 승일경우 무조건 양수 값을 가지기 때문이다. 마지막 5단계에서는 고정소수점으로 연산된 결과를 부동소수점 포맷으로 바꿔준다. Zero leading counter(ZLC)를 이용하여 지수영역의 크기를 결정해주며, normalizer을 이용하여 가수영역을 결정한다.

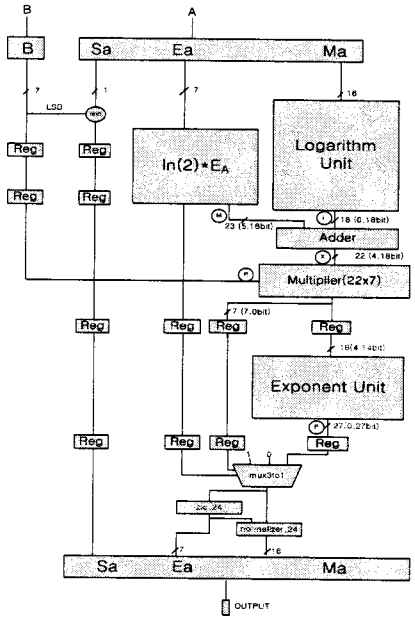

그림 1. 멱승기의 구조

3.1 로그연산기의 구현

로그연산기는 가수영역 M_A을 입력으로 사용하며, 히든비트인 1비트를 합하여 뉴튼 방식의 D를 나타낸다. 그림 2는 로그연산기의 구조를 나타낸 것이다. 초기값 x_0을 구하기 위한 룸 테이블의 입력으로는 소수 11비트만 사용된다. 히든비트인 정수 1비트는 모든 입력에 동일하게 적용되기 때문에 테이블의 입력값에는 반영되지 않는다. 이는 테이블 사이즈를 줄이기 위한 방법이다. EXP_LUT는 지수연산을 위한 룸 테이블이다. 2개의 테이블을 통해 1번의 반복연산에 사용될 데이터들을 모두 구하며, 이를 레지스터에 저장한다. 이후 곱셈기와 덧셈기를 이용하여 뉴튼 방식을 이용한 한번의 반복연산을 수행한다. 2cycles 이후 로그연산의 최종 결과값인 L이 생성된다. 전체 로그연산에 사용되는 룸 테이블 사이즈는 로그연산 테이블 $9x2^{11}$과 지수연산 테이블 $18x2^{9}$이며, 전체 27k비트가 사용되었다.

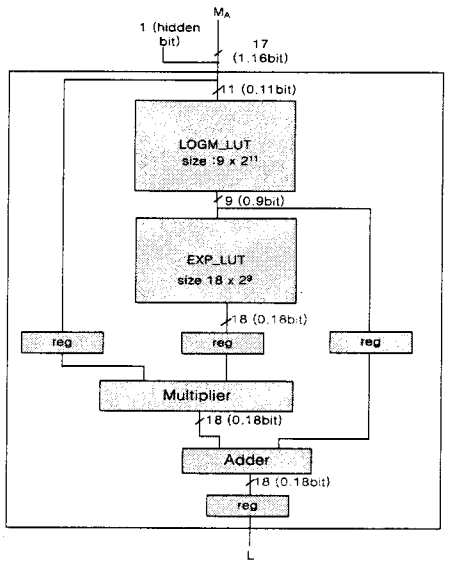

그림 2. 로그연산기의 구조

3.2 지수 연산기

지수연산기는 정수 4비트, 소수 14비트를 입력으로 받는다. 지수연산기의 구조는 아래 그림 3과 같다. 룸 테이블 사이즈를 줄이기 위해 입력값을 이분하여 처리한다. 룸 테이블의 사이즈는 $2x(16x2^9)$비트이다. 하지만 EXP_LUT2은 일정 값 이상이 되면 0으로 수렴하므로 모든 입력에 대해 저장하지 않는다. EXP_LUT2은 약 5.5k비트의 크기를 갖는다. 결과적으로 지수연산기에 사용된 룸 테이블의 크기는 약 13.5k비트이며, 1cycle의 딜레이 타임을 갖는다.

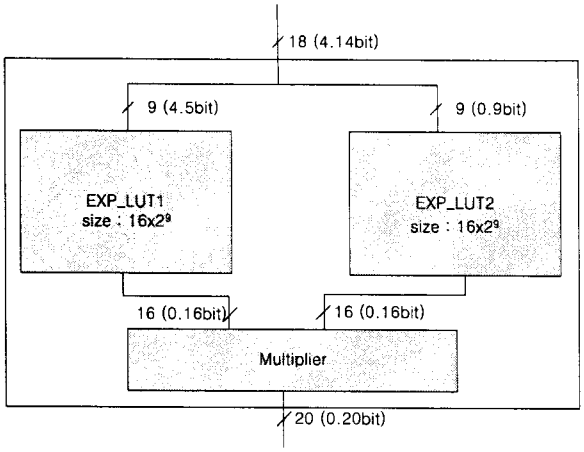

그림 3. 지수연산기의 구조

4. 결 론

본 논문에서는 모바일용 3D 그래픽 엔진에서 사용되는 멱승기를 제안하였다. 일반적으로 지수연산은 매우 복잡하며, 많은 자원을 요구하는데, 본 논문의 구조는 최대한 자원을 절약하면서 빠른 연산을 수행 할 수 있도록 제안되었다. 연산 소요시간은 5 cycles 이며, 파이프 라인 구조로서 연속적인 연산이 가능하다. 합성 결과는 Magnachip 0.18μm 공정에서 100Mhz의 동작주파수를 만족하였으며, 약 16k gates을 차지하였다. 입력값 A, B에 대해 10^{-4}의 정확도를 만족하였다. 정확도는 C code을 이용한 시뮬레이션을 통하여 검증하였다.

감사의 글

본 논문의 IDEC의 부분지원으로 이루어졌습니다.

본 논문은 정보통신부 출연금으로 ETRI, SoC 산업진흥센터에서 수행한 IT SoC 핵심설계인력양성사업의 부분 지원으로 이루어진 연구결과입니다.

본 논문의 내용을 발표할 때에는 반드시 ETRI, SoC 산업진흥센터, IT SoC 핵심설계인력양성사업의 부분 지원으로 이루어진 결과임을 밝혀야 합니다.

참 고 문 헌

[1] Tomas Akenine-Mollar, Eric Hanines, "Real-Time Rendering 2nd," A K Peters, 2002.

[2] 권용국, 김기철, "휴대단말기용 3차원 그래픽 프로세서를 위한 제산기 설계," *대한전자공학회 SoC 학술대회 논문집*, 166-169 쪽, 숭실대학교 2006년 5월.

[3] Brubaker, T.A, and Becker, J.C, "Multiplication using logarithms implemented with read-only memory," IEEE Trans, COM-24, Issue 8, pp761-765, 1975.

[4] Kioutelidis, J.B., and Petrou, J.K, "A piecewise linear approximation of log_2x with equal maximum errors in all intervals," IEEE Trans, COM-24, Issue 9, pp.858-861, Sept. 1975.

[5] Zhang, M, Delgado-Frias, J.G, and Vassiliadis, S. "Table driven Newton scheme for high precision logarithm generation," *IEEE Trans. Computers,* volume141, no. 5, pp. 281-292, Sept. 1994.