ISSN: 1226-7244 (Print)
ISSN: 2288-243X (Online)
논문번호 13-03-18
126

Journal of IKEEE.Vol.17,No.3,346~351,September 2013
http://dx.doi.org/10.7471/ikeee.2013.17.3.346

Division-by-Convergence 방식을 사용하는 24-비트 부동소수점 제산기에 대한 OpenGL 정확도의 대수적 검증

Algebraic Accuracy Verification for Division-by-Convergence based 24-bit Floating-point Divider Complying with OpenGL

유 세 훈*, 이 정 우*, 김 기 철*★

Sehoon Yoo*, Jungwoo Lee*, Kichul Kim*★

Abstract

Low-cost and low-power are important requirements in mobile systems. Thus, when a floating-point arithmetic unit is needed, 24-bit floating-point format can be more useful than 32-bit floating-point format. However, a 24-bit floating-point arithmetic unit can be risky because it usually has lower accuracy than a 32-bit floating-point arithmetic unit. Consecutive floating-point operations are performed in 3D graphic processors. In this case, the verification of the floating-point operation accuracy is important. Among 3D graphic arithmetic operations, the floating-point division is one of the most difficult operations to satisfy the accuracy of 10^{-5} which is the required accuracy in OpenGL ES 3.0. No 24-bit floating-point divider, whose accuracy is algebraically verified, has been reported. In this paper, a 24-bit floating-point divider is analyzed and it is algebraically verified that its accuracy satisfies the OpenGL requirement.

요 약

모바일 시스템에서는 비용 및 전력 효율이 중요하기 때문에 부동소수점 연산기 개발 시 32-비트 데이터 형식 대신 24-비트 데이터 형식을 사용하는 것이 좋다. 하지만 24-비트 데이터 형식을 사용할 경우 32-비트 데이터 형식에 비해 연산기의 정확도가 낮아질 수 있다. 3D 그래픽과 같이 연속적인 부동소수점 연산 처리가 많이 요구될 경우 연산기의 정확도에 대한 논의와 검증이 중요하다. 나눗셈은 3D 그래픽에 사용되는 연산 중 OpenGL에서 규정한 정확도를 만족하기 가장 어려운 연산 중 하나이다. 현재까지 OpenGL에서 규정한 정확도를 만족하는 것이 대수적으로 검증된 24-비트 부동소수점 제산기는 알려진 바가 없다. 본 논문에서는 24-비트 부동소수점 제산기를 분석하고, OpenGL ES 3.0에서 규정한 10^{-5}의 정확도를 만족함을 대수적으로 검증한다.

Key words : OpenGL ES 3.0, 24-bit floating-point divider, division-by-convergence, accuracy, algebraic verification.

* Dept. of Electrical and Computer Engineering, University of Seoul,
yugdoog@uos.ac.kr, 02-6490-5690

★ Corresponding author

※ This work was supported by Ministry of Trade, Industry and Energy(MOTIE) and IDEC Platform Center(IPC).

Manuscript received Aug. 30, 2013; revised Sep. 11, 2013; accepted Sep. 16. 2013

Ⅰ. 서론

OpenGL은 3D 그래픽스에서 널리 사용되고 있는 국제표준이다. 임베디드 시스템에 특화된 OpenGL ES에서는 3D 그래픽스의 기본 단위 연산인 부동소수점 연산의 정확도에 대해 오차의 최대치를 10^{-5}로 규정하고 있다[1]. 3D 그래픽스에서는 많은 양의 부동소

수점 연산이 요구된다. 이 때, 3D 그래픽 처리 과정의 많은 연산으로 인해 누적된 오차는 화질에 직접적인 영향을 미치기 때문에 OpenGL에서 규정한 부동소수점 연산의 정확도는 매우 중요한 요소이다. OpenGL ES 3.0 표준 문서에서는 10^{-5}의 정확도가 32-비트 부동소수점 연산 방법에서 대부분 만족된다고 언급되어 있다[1]. 정확도를 만족하기 위해 32-비트나 64-비트의 부동소수점 연산을 사용해도 PC, 워크스테이션, 콘솔 게임 등의 그래픽 환경에서는 큰 문제가 되지 않는다. 반면, 자원이 부족한 모바일 그래픽 환경에서는 전력과 면적을 최소화하는 것이 중요하기 때문에 저전력/저면적을 사용하는 3D 그래픽 처리 하드웨어에 대한 연구가 필요하다. 이에 대해 log 도메인으로 변경하여 고정소수점 연산을 수행하는 방법과 24-비트 부동소수점 형식을 사용하는 방법이 연구되어 있다[2]-[7]. 이 중 본 논문에서는 24-비트 부동소수점 형식을 사용하는 방법에 초점을 맞췄다. 24-비트 부동소수점 형식을 사용하면 32-비트 부동소수점에 비해 저전력/저면적의 특징을 가질 수 있다[2]-[5]. 하지만 24-비트 데이터 형식의 경우 32-비트 데이터 형식에 비해 정확도가 낮아질 수 있다는 위험성이 있다. 따라서 24-비트 부동소수점 연산기 설계 시 정확도에 대한 심도있는 논의와 검증이 필요하다.

지금까지 24-비트 부동소수점 형식을 사용하는 연산기들에 대한 연구가 수행되어 있다[3]-[5]. 이 연구들에서는 OpenGL에서 규정한 10^{-5}의 정확도를 만족하는 것을 검증하지 않거나, 대부분의 경우에 대해 정확도를 만족함을 실험적으로 보였다. 본 논문에서는 24-비트 부동소수점 제산기[5]의 구조를 사용하여 제산기의 오차를 대수적으로 분석한다. 대수적 오차 분석을 통해 존재하는 모든 경우에 대해서 OpenGL 정확도를 만족한다는 것을 검증한다.

본 장의 나머지 부분은 다음과 같이 구성된다. 2장에서는 본 논문의 제산기에 사용된 나눗셈 알고리듬인 Division-By-Convergence를 소개한다. 3장에서는 24-비트 부동소수점 포맷과 이를 이용한 부동소수점 제산기의 구조를 살펴본다. 또한 부동소수점 제산기에 대한 대수적 오차 분석을 설명한다. 마지막으로 4장에서 본 논문의 결론을 맺는다.

II. Division-By-Convergence

3D 그래픽에서 나눗셈은 중요한 연산이다. 3D 그래픽 처리 과정에서 실제 나눗셈 연산의 횟수가 많지는 않지만, 나눗셈의 연산 속도가 느릴 경우 3D 그래픽 시스템 전체에 병목 현상이 야기될 수 있기 때문에 나눗셈 연산의 중요도는 매우 높다. 병목 현상을 피하고 성능 향상을 위해서는 하드웨어 구현에 적합한 고속의 나눗셈 알고리듬이 필요하다.

지금까지 다양한 나눗셈 알고리듬이 연구되었다[8]-[10]. 그 중에서 Division-By-Convergence 알고리듬은 잘 알려진 나눗셈 알고리듬으로써 곱셈기를 사용한 고속 제산기에 적합하다. 아래에 Division-By-Convergence 알고리듬을 소개한다[8].

$$Q=\frac{N}{D}=\frac{N\bullet R_0R_1\cdots R_{n-1}}{D\bullet R_0R_1\cdots R_{n-1}}=\frac{Q}{1}\approx\frac{Q'}{0.11\cdots1} \tag{1}$$

식(1)에서 Q는 나눗셈의 몫, N은 분자(또는 피제수), D는 분모(또는 제수)이다. D는 $0.5\le D<1$의 범위를 갖는다. 여기에서 분모와 분자에 곱셈 요소 $R_k, k=0,1,\cdots,n-1$을 계속 곱하여 분모가 1에 수렴(convergence)하도록 하면 결국 분자에 몫이 남게 된다. 이 때, 알고리듬에서는 분모가 1이 아니라 0.11...1로 수렴하게 되지만 소수점 뒤에 충분한 양의 1이 있을 경우 실제 몫 Q에 가까운 알고리듬 몫 Q'를 얻게 된다. Division-By-Convergence 알고리듬에서는 곱셈 요소 R_k를 구할 방법이 필요하다. 이를 위해 아래의 관계식을 보인다.

$$D_{k+1}=D_kR_k=(1-a^{2^k})(1+a^{2^k})=1-a^{2^{(k+1)}} \tag{2}$$

식(2)는 k-단계에서 분모 D_x와 곱셈 요소 R_k를 곱했을 때 생성되는 $(k+1)$-단계의 새로운 분모 D_{k+1}를 보여준다. 여기에서 a는 $0<a\le\frac{1}{2}$의 범위를 갖는 수이다. 식(2)에 근거하여 가장 처음 단계에서 D에 R_0을 곱했을 경우를 표현하면 식(3)과 같다.

$$D_1=DR_0=(1-a)(1+a)=1-a^2 \tag{3}$$

이 때, $0<a^2\le\frac{1}{4}$이므로 $\frac{3}{4}\le1-a^2<1$가 성립된다. 따라서 $D_1=0.11xx\ldots x$가 된다. 다음 단계는 식(4)와 같다.

$$D_2=D_1R_1=(1-a^2)(1+a^2)=1-a^4 \tag{4}$$

여기에서 $0 < a^4 \le \frac{1}{16}$이므로, $D_2 = 0.1111xx\ldots x$가 된다. 비슷한 방법으로 곱셈을 계속 진행하면 D_{n-1}는 $2^{(n-1)}$개의 연속적인 1을 가지게 된다. 곱셈 요소 R_k는 식(2)에 근거하여 식(5)처럼 구해질 수 있다.

$$R_k = 1 + a^{2^k} = 2 - (1 - a^{2^k}) = 2 - D_k \quad (5)$$

식(5)에 보인 것처럼 R_k는 같은 단계의 분모 D_k로부터 계산될 수 있다.

III. 24-비트 부동소수점 제산기의 오차 분석

본 논문에서 사용하는 24-비트 부동소수점 형식은 그림 1과 같다[5]. 사용된 24-비트 부동소수점 형식은 IEEE 754 단정도 형식과 유사한 형식을 가진다[11]. 부호비트, 지수비트, 가수비트로 나뉘어 있으며, 가수부분은 hidden 비트를 사용한다.

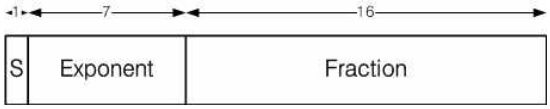

Fig. 1. 24-bit floating-point data format
그림 1. 24-비트 부동소수점 데이터 형식

본 논문에서 사용하는 24-비트 부동소수점 제산기는 그림 2와 같다. 그림에 보인 제산기는 Y. Kweon 등이 제안한 연산기[5]를 보완한 것으로, Division-By-Convergence 알고리듬[8]과 ROM 테이블을 이용하여 빠른 연산이 가능하다. 본 논문에서는 OpenGL ES 3.0에 규정된 정확도를 맞추기 위해 첫 번째 파이프라인 단계에서의 truncation되는 비트 크기와 두 번째 파이프라인 단계에서 사용된 곱셈기의 크기가 수정되었다. *S_N*, *E_N*, *M_N*은 각각 분자의 부호비트, 지수비트, 가수비트이다. *S_D*, *E_D*, *M_D*은 각각 분모의 부호비트, 지수비트, 가수비트이다. 이 부동소수점 제산기에 대한 오차 분석을 위해 본 논문에서는 주로 점선 안에 포함되는 가수비트 처리 부분을 살펴본다.

제산기의 오차는 크게 제산기의 구조에서 발생되는 구조적 오차와 Division-By-Convergence 알고리즘에서 발생하는 알고리듬 오차로 나눌 수 있다.

3.1 부동소수점 제산기의 구조적 오차

그림 2에서 비정규화(denormalization) 이전의 분모와 분자의 가수비트를 각각 *M_N*, *M_D*이라 하고, 비정규화 과정을 거친 값을 각각 *N*, *D*라 가정한다. *N*, *D*는 비정규화 과정을 거쳤기 때문에 $1 \le N, D < 2$의 범위를 갖는다. *N*과 *D*를 2로 나눈 값을 각각 *N'*와 *D'*라 하면 식(6)과 같이 나타낼 수 있다.

$$Q = \frac{N}{D} = \frac{N/2}{D/2} = \frac{N'}{D'} = \frac{Q'}{0.11\ldots1} \approx \frac{Q}{1} \quad (6)$$

N', *D'*는 $0.5 \le N', D' < 1$의 범위를 가지며, *N'*, *D'*을 입력으로 사용하여 Division-By-Convergence 알고리듬을 통해 몫 *Q'*를 구하게 된다.

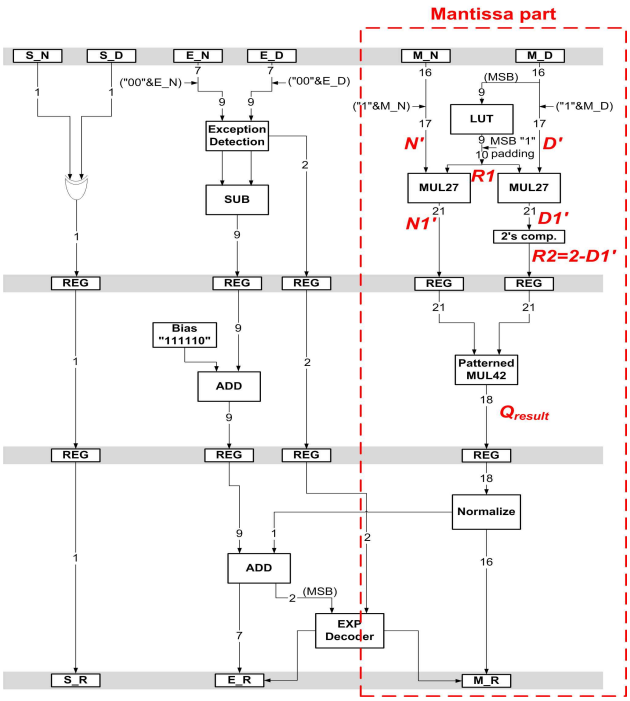

Fig. 2. A block diagram of the floating-point divider
그림 2. 부동소수점 제산기의 블록도

그림 2에서 부동소수점 제산기는 세 개의 파이프라인 단계를 갖는다. 첫 번째 파이프라인 단계에서는 LUT (look-up table)에서 출력된 곱셈 요소 *R*1과 *N'*, *D'*의 곱셈이 이루어진다. 곱셈 요소 *R*1은 *D'*과 곱했을 때 나오는 출력 *D*1'가 연속적인 1을 최소 9개 갖도록 LUT에서 선택된다. 따라서 LUT를 사용함으로써 Division-By-Convergence 알고리듬의 수행 단계를 줄일 수 있다. LUT의 결과값이 곱셈기에서 사용될 때 분모와 분자에 동일한 값이 곱해지므로 몫을 구할 때 LUT 사용으로 인한 오류는 발생하지 않는다

[8]. 두 번째 파이프라인 단계를 거치면 분모는 연속적인 1을 최소 18개 갖게 된다. 세 번째 단계에서는 이전 파이프라인 단계의 결과값을 24-비트 부동소수점 형식에 맞추기 위해 정규화 과정이 수행된다[5].

첫 번째 파이프라인 단계에서 사용하는 27-비트 곱셈기는 결과 값의 상위 21-비트만 출력하고 나머지 비트는 truncation 되기 때문에 정보 손실이 발생된다. $R1$과 N'의 곱셈 결과로 나오는 새로운 분자를 $N1$, truncation 이후의 분자를 $N1'$, $R1$와 D'의 곱셈 결과로 나오는 새로운 분자를 $D1$, truncation 이후의 분모를 $D1'$라 가정한다. 식(6)을 다시 나타내면 식(7)과 같다.

$$Q = \frac{N}{D} = \frac{N'}{D'} = \frac{N' \cdot R1}{D' \cdot R1} = \frac{N1}{D1} \tag{7}$$

분자와 분모에 대해 truncation으로 인하여 발생되는 오차를 각각 e_N, e_D라 하면, 식(8)과 같이 나타낼 수 있다.

$$\begin{aligned} N1' &= N1 - e_N \\ D1' &= D1 - e_D \end{aligned} \tag{8}$$

Truncation된 하위 6-비트는 2^{-22}자리부터 2^{-27}자리까지기 때문에 truncation으로 인한 e_N, e_D의 범위는 식(9)와 같다.

$$0 \le e_N, e_D < 2^{-21} \tag{9}$$

식(8)을 대입하면 식(7)은 식(10)과 같이 나타낼 수 있다.

$$Q = \frac{N}{D} = \frac{N1}{D1} = \frac{N1' + e_N}{D1' + e_D} \tag{10}$$

그림 2에서 $D1'$의 값에 대해 2의 보수로 취하면 $R2$의 값을 얻을 수 있다. $R2$은 두 번째 파이프라인 단계에서 $N1'$의 값과 곱해지게 된다. 이 곱셈의 결과는 식(11)과 같이 나타낼 수 있다.

$$\frac{Q_{result}}{0.11\ldots1} = \frac{N1' \cdot R2}{D1' \cdot R2} = \frac{N1'(2 - D1')}{D1'(2 - D1')} \tag{11}$$

식(11)의 Q_{result}는 정규화(normalization) 과정 이전의 나눗셈 결과값으로 truncation에 인한 오차가 포함된 값이다. 식(8)을 식(11)에 대입하면 식(12)와 같이 나타낼 수 있다.

$$\begin{aligned} \frac{Q_{result}}{0.11\ldots1} &= \frac{N1'(2 - D1')}{D1'(2 - D1')} \\ &= \frac{(N1 - e_N)(2 - D1 + e_D)}{(D1 - e_D)(2 - D1 + e_D)} \\ &= \frac{N1(2 - D1) + N1 \cdot e_D - e_N(2 - D1) - e_D e_N}{0.11\ldots1} \end{aligned} \tag{12}$$

식(13)은 truncation에 의한 오차가 없을 경우의 나눗셈 결과값을 나타낸 것이다.

$$\begin{aligned} Q &= \frac{N' \cdot R1 \cdot R2}{D' \cdot R1 \cdot R2} = \frac{N1 \cdot R2}{D1 \cdot R2} = \frac{N1(2 - D1)}{D1(2 - D1)} \\ &= \frac{Q'}{0.11\ldots1} \end{aligned} \tag{13}$$

식(12)의 오차가 포함된 몫 Q_{result}와 식(13)의 오차가 없는 몫 Q'의 관계식은 식(14)처럼 나타낼 수 있다.

$$Q_{result} = Q' + E_S \tag{14}$$

식(14)에서 E_S는 truncation에 의해서 발생된 오차이다. 식(13)에서 $Q' = N1(2 - D1)$이므로 식(12)에서 Q_{result}를 다시 표현하면 식(15)와 같다.

$$\begin{aligned} Q_{result} &= N1(2 - D1) + N1 \cdot e_D - e_N(2 - D1) - e_D e_N \\ &= Q' + E_1 + E_2 + E_3 = Q' + E_S \\ &\text{단, } E_1 = N1 \cdot e_D,\ E_2 = -e_N(2 - D1),\ E_3 = -e_D e_N, \\ &E_S = E_1 + E_2 + E_3 \end{aligned} \tag{15}$$

아래는 각 변수에 대한 범위를 정리한 것이다.

$$\begin{aligned} 0.5 \le N', D' < 1 \\ 1 \le R1 < 2 \\ 0 \le e_N, e_D < 2^{-21} \end{aligned}$$

이를 이용하여 E_1, E_2, E_3항에 대한 오차 범위를 구하면 다음과 같다.

1) E_1

$$\begin{aligned} & 0.5 \le N' < 1 \\ \Rightarrow\ & 0.5 \le N' R1 < 2 \\ \Rightarrow\ & 0 \le N' R2\, e_D < 2^{-20} \\ \Rightarrow\ & 0 \le E_1 < 2^{-20} \\ & (\because E_1 = N1 \cdot e_D = N' R2\, e_D) \end{aligned} \tag{16}$$

2) E_2

$$\begin{aligned} & 0.5 \le D' < 1 \\ \Rightarrow & 0.5 \le D'R_0 < 2 \\ \Rightarrow & 0 < 2 - D'R_0 \le 1.5 \\ \Rightarrow & 0 \le e_N(2 - D'R_0) < 2^{-21} + 2^{-22} \\ \Rightarrow & -(2^{-21} + 2^{-22}) < -e_N(2 - D'R_0) \le 0 \\ \Rightarrow & -(2^{-21} + 2^{-22}) < E_2 \le 0 \\ & (\because E_2 = -e_N(2 - D_1) = -e_N(2 - D'R_0)) \end{aligned} \quad (17)$$

3) E_3

$$\begin{aligned} & 0 \le e_D e_N < 2^{-42} \\ \Rightarrow & -2^{-42} < -e_D e_N \le 0 \\ \Rightarrow & -2^{-42} < E_3 \le 0 \\ & (\because E_3 = -e_D e_N) \end{aligned} \quad (18)$$

따라서 오차의 합 $E_1 + E_2 + E_3$의 오차범위는 식(19)와 같다.

$$-(2^{-21} + 2^{-22} + 2^{-42}) < E_1 + E_2 + E_3 < 2^{-20} \quad (19)$$

3.2 부동소수점 제산기의 알고리듬 오차

식(11)에서 Q와 Q' 사이에는 오차가 존재한다. 이 오차는 Division-By-Convergence 알고리듬에서 발생되는 알고리듬 오차이다. 알고리듬 오차를 E_A라 하면 식(20)과 같이 나타낼 수 있다. 본 논문에서는 분모에서 18개의 연속적인 1을 보장하기 때문에 E_A는 $0 < E_A \le 2^{-18}$의 범위를 갖는다.

$$\begin{aligned} & Q = \frac{Q'}{0.11...1} = \frac{Q'}{1 - E_A} \\ & (0 < E_A \le 2^{-18}) \end{aligned} \quad (20)$$

3.3 부동소수점 제산기의 전체 오차

본 논문에서 사용한 부동소수점 제산기의 전체 오차를 구하기 위해서 식(20)에서 식(14)를 대입하면 식(21)과 같이 나타낼 수 있다.

$$\begin{aligned} & Q = \frac{Q'}{1 - E_A} = \frac{Q_{result} - E_S}{1 - E_A} \\ & Q(1 - E_A) = Q_{result} - E_S \\ & Q - QE_A = Q_{result} - (E_1 + E_2 + E_3) \\ & Q - Q_{result} = QE_A - (E_1 + E_2 + E_3) \end{aligned} \quad (21)$$

식(21)에서 $Q - Q_{result}$는 실제 이론적인 몫의 값과 부동소수점 제산기의 출력 Q_{result} 사이의 최종 오차를 의미한다.

D'와 N'의 범위가 $0.5 \le N', D' < 1$이므로 식(6)에서 Q는 $0.5 < Q < 2$의 범위를 갖는다. 만일 Q의 범위가 $0.5 < Q < 1$일 경우 그림 2의 세 번째 파이프라인 단계의 normalization 과정에서 왼쪽으로 1-비트를 시프트하여 $1 < Q < 2$의 범위가 되도록 한다. Q의 범위를 $0.5 < Q < 1$와 $1 \le Q < 2$의 두 구간으로 구분하여 오차범위를 구하면 다음과 같다.

1) $0.5 < Q < 1$

$$\begin{aligned} & 0.5 < Q < 1 \\ \Rightarrow & 0 < QE_A < 2^{-18} \ (\because \text{식(20)}) \\ \Rightarrow & -2^{-20} < QE_A - (E_1 + E_2 + E_3) \\ & < (2^{-18} + 2^{-21} + 2^{-22} + 2^{-42}) \ (\because \text{식(19)}) \\ \Rightarrow & -0.0000010 < QE_A - (E_1 + E_2 + E_3) \\ & < 0.0000046 \\ \Rightarrow & -0.0000020 < 2[QE_A - (E_1 + E_2 + E_3)] \\ & < 0.0000092 \ (\because \text{왼쪽으로 1비트 시프트}) \end{aligned} \quad (22)$$

2) $1 \le Q < 2$

$$\begin{aligned} & 1 \le Q < 2 \\ \Rightarrow & 0 < QE_A < 2^{-17} \\ \Rightarrow & -2^{-20} < QE_A - (E_1 + E_2 + E_3) \\ & < (2^{-17} + 2^{-21} + 2^{-22} + 2^{-42}) \\ \Rightarrow & -0.0000010 < QE_A - (E_1 + E_2 + E_3) \\ & < 0.0000084 \end{aligned} \quad (23)$$

식(22), 식(23)은 부동소수점 제산기의 전체 오차 범위를 나타내고 있다. Q의 범위가 $0.5 < Q < 1$와 $1 \le Q < 2$일 경우 모두 OpenGL ES 3.0에서 요구하는 오차 범위 10^{-5}를 만족한다.

Ⅳ. 결론

현재까지 24-비트 부동소수점 형식을 사용하는 연산기들이 연구되었지만, OpenGL ES 3.0의 정확도 요구사항인 10^{-5}를 만족하는 것을 대수적으로 검증한 연구는 알려진 바가 없다. 3D 그래픽의 연속적인 부동소수점 연산을 고려했을 때 부동소수점 연산기에 발생하는 오차는 전체 3D 그래픽 처리 결과에 적지 않은 영향을 줄 수 있다. 따라서 부동소수점 연산기의 오차 분석이 매우 중요하다. 본 논문에서는 24-비트 부동소수점 제산기의 구조를 사용하여 대수적으로 오차를 분석하였다. 이를 통해 이 제산기가 OpenGL ES에서 규정한 10^{-5} 정확도를 만족하는 것을 검증하

였다. 본 논문에서 대수적으로 오차가 검증된 부동소수점 제산기는 모바일 시스템에서 신뢰도 높은 3D 그래픽 시스템을 설계하는 데에 사용될 수 있다.

References

[1] B. Lipchak, "OpenGL ES Version 3.0," Khronos Group Inc., Aug. 2012
[2] K. Y. Lee, "A design of a floating point unit with 3 stages for a 3D graphics shader engine," Journal of IKEEE, vol. 11, no. 4, pp. 358-363, Dec. 2007.
[3] J. S. Ha, H. G. Jeong, S. Y. Kim, and K. Y. Lee,"Design of a 3D graphics geometry accelerator using the programmable vertex shader," Journal of IEEK SD, vol. 43, no. 9, pp. 53-58, Sep. 2006.
[4] C. Lee, "Design of square root and inverse square root arithmetic units for mobile 3D graphic processing," Journal of IEEK SD, vol. 46, no. 3, pp. 20-25, March 2009.
[5] Y. Kweon and K. Kim, "A floating-point divider for mobile 3D graphics processors," International SoC Conference (ISOCC), pp. 545-548, Oct. 2006.
[6] F. Sheikh, S. K. Mathew, M. A. Anders, H. Kaul, S. K. Hsu, A. Agarwal, R. K. Krishnamurthy, and S. Borkar, "A 2.05 GVertices/s 151 mW lighting accelerator for 3D graphics vertex and pixel shading in 32 nm CMOS," IEEE Journal of Solid-State Circuits, vol. 48, no. 1, pp. 128-139, Jan. 2013.
[7] S. Hsiao, P. Wu, C. Wen, and L. Chen, "Design of a programmable vertex processor in OpenGL ES 2.0 mobile graphics processing units," International Symposium on VLSI Design, Automation, and Test (VLSI-DAT), pp. 1-4, April 2013.
[8] I. Koren, Computer Arithmetic Algorithms 2nd ed., A K Peters, Massachusetts, Nov. 2001.
[9] N. Louvet, J. M. Muller, and A. Panhaleux, "Newton-Raphson algorithms for floating-point division using an FMA," IEEE International Conferfence on Application-specific Systems Architectures and Processors (ASAP), pp. 200-207, July 2010.
[10] D. M. Russinoff, "Computation and formal verification of SRT quotient and square root digit selection tables," IEEE Transactions on Computers, vol. 62, no. 5, pp. 900-913, May 2013.
[11] ANSI/IEEE std 7544-1985, "IEEE Standard for Binary Floating-Point Arithmetic," 1985.

BIOGRAPHY

Sehoon Yoo (Student Member)

2006 : BS degree in Electrical and Computer Engineering, University of Seoul.
2008 : MS degree in Electrical and Computer Engineering, University of Seoul.
2008~ : Working toward the PhD degree in Electrical and Computer Engineering, University of Seoul.

Jungwoo Lee (Member)

2005 : BS degree in Electrical and Computer Engineering, University of Seoul.
2007 : MS degree in Electrical and Computer Engineering, University of Seoul.
2012 : PhD degree in Electrical and Computer Engineering, University of Seoul.
2012~ : Senior Research Engineer, Chief Technology Office (CTO), LG Electronics.

Kichul Kim (Member)

1982 : BS degree in Electrical Engineering, Seoul National University.
1984 : MS degree in Electrical Engineering, Seoul National University.
1991 : PhD degree in Electrical Engineering, University of Southern California.
1984~1994 : Senior Research Engineer, Electronics and Telecommunications Research Institute (ETRI).
1994~ : Professor, in Electrical and Computer Engineering, University of Seoul.