Journal of Korea Multimedia Society Vol. 22, No. 9, September 2019(pp. 1029-1035)
https://doi.org/10.9717/kmms.2019.22.9.1029

부동소수점수 N차 제곱근 K차 골드스미스 알고리즘

조 경 연†

Floating Point Number N'th Root K'th Order Goldschmidt Algorithm

Gyeong Yeon Cho†

ABSTRACT

In this paper, a tentative Kth order Goldschmidt floating point number Nth root algorithm for K order convergence rate in one iteration is proposed by applying Taylor series to the Goldschmidt square root algorithm. Using the proposed algorithm, Nth root and Nth inverse root can be computed from iterative multiplications without division. It also predicts the error of the algorithm iteration. It iterates until the predicted error becomes smaller than the specified value. Since the proposed algorithm only performs the multiplications until the error gets smaller than a given value, it can be used to improve the performance of a floating point number Nth root unit.

Key words: Floating Point Number Nth Root, Kth Order Goldscjhmidt, Square Root, Cubic Root

1. 서 론

지난 수십년간 반도체의 집적도와 동작속도가 빨라지면서 SoC(System on Chip)의 기능과 성능이 비약적으로 발전하였다. 여러 개의 마이크로프로세서와 멀티미디어 시스템, 무선통신 등의 다양한 기능을 하나의 칩에 집적하는 AP(Application Processor)가 보편화되고 있다. AP 등 멀티코어 구조에서 각 프로세서들은 동일한 명령어를 수행하지만 특정 연산의 하드웨어 가속 기능을 별도로 가지는 것이 적합하다[1]. 멀티미디어 시스템, 디지털 신호처리, 무선 통신, 과학기술연산, 시뮬레이션 등을 효율적으로 처리하기 위해서는 다양한 연산 기능이 필요하다[2]. 부동소수점수 Z의 N차 제곱근($\sqrt[N]{Z}$)은 컴퓨터그래픽스 등 멀티미디어 시스템, 과학 및 공학 기술 분야에서 많이 사용된다.

N차 제곱근을 구하는 문제는 오래 된 문제로 한번에 한 디지트씩 연산하는 방식[3,4,5]과 반복 연산으로 근에 수렴하는 방식이 있다. 곱셈을 반복하여 나눗셈과 제곱근을 구하는 방식으로는 뉴톤-랍손 방식, 골드스미트 방식 등이 있다. 이들은 곱셈을 반복하여 2차 수렴(quadratic convergence)하는 근을 구한다[6]. 이들을 3차, 4차 등 고차 수렴으로 제곱근을 구하는 연구들이 있다[7]. N차 제곱근은 뉴톤-랍손 방식을 확장하는 방식[8,9]과 할리 방식[10] 등이 있다. 이들 방식들은 나눗셈 또는 역수 계산이 필요하므로 CPU에서 나눗셈과 곱셈의 연산속도가 동일하거나 크게 차이나지 않는다는 가정을 근거로 하고 있다[7]. 그러나 현재의 기술로는 부동소수점 나눗셈은 곱셈에 비하여 대단히 느리다[12].

종래의 골드스미트 제곱근 알고리즘은 초기 근사값과 최대 오차를 계산하고, 오차가 정해진 값보다 작게 될 때까지 반복 연산을 수행하였다. 종래는 최대 오차만을 고려했기 때문에, 결과 값에 도달했음에

※ Corresponding Author : Gyeong Yeon Cho, Address: (48513) :Pukyong National University. 45, Yongso-ro, Nam-Gu. Busan, Korea, TEL : +82-10-9081-8231, FAX : +82-51- 629-6210, E-mail : gyuncho@hanmail.net
Receipt date : Apr. 10, 2019, Revision date : Aug. 19, 2019
Approval date : Sep. 2, 2019
† Department of IT Convergence and Application Engineering, Pukyong National University
※ This work was supported by a Research Grant of Pukyong National University(2019)

도 불구하고 가외의 연산을 수행하여 연산 속도를 저하시키는 단점이 있었다. [13,14]은 가변 시간 알고리즘을 제안하여 이러한 문제를 개선했다. [15]는 테일러급수 정리로부터 골드스미트 K차 나눗셈 알고리즘을 제안하였다. K차 수렴 나눗셈 알고리즘은 한 번 반복에 K번 곱셈을 수행한다. 또한 알고리즘 반복 과정의 오차를 예측하고, 예측한 오차가 정해진 값보다 작아지는 시점까지만 반복 수행한다.

본 논문에서는 골드스미트 제곱근 알고리즘에 테일러급수를 적용하여 곱셈만을 사용하여 N차 제곱근을 구하는 가칭 N차 제곱근 K차 골드스미트 알고리즘을 제안한다.

본 논문에서는 N차 제곱근과 N차 역제곱근을 구하는 두 가지 알고리즘을 제안한다. 또한 다음 반복 연산에서의 오차를 예측하여 가변시간 연산이 가능하도록 한다.

본 논문의 구성은 다음과 같다. 2장에서는 N차 제곱근을 연산하는 알고리즘을 제안한다. 3장에서는 N차 역제곱근을 연산하는 알고리즘을 제안한다. 4장에서 연구 결과를 종래의 골드스미트 알고리즘과 비교한다. 5장에서 결론을 맺는다.

2. N차 제곱근 K차 골드스미트 알고리즘

2.1 N차 제곱근 K차 골드스미스 알고리즘

부동소수점수는 $Z=1.f_2\times 2^p$으로 표현된다. 가수부분과 지수 부분을 분리해서 N차 제곱근을 구하면 식 (1)이 된다.

$$\sqrt[N]{Z}=\sqrt[N]{1.f\times 2^{p \bmod N}}\times 2^{(\frac{p}{N})} \tag{1}$$

식 (1)에서 $(\frac{p}{N})$은 나눗셈 결과의 정수부분만을 취한다. 근의 수렴속도와 오차는 $1.f$에 비례하므로 본 논문에서는 식 (2)로 Z의 N차 제곱근을 구한다. 또한 간단한 표기를 위하여 $1.f$를 F로 표기한다.

$$\sqrt[N]{Z}=\sqrt[N]{F}\sqrt[N]{2^{p \bmod N}}\times 2^{(\frac{p}{N})} \tag{2}$$

부동소수점 수 $F=1.f$의 N차 제곱근 $Y_n=\sqrt[N]{1.f}$은 초기값 Y_0를 정의하고, 반복식으로 $Y_i(i=1,...,n)$을 구한다. IEEE-754로 규정되는 부동소수점 수에서 가수부 $1.f_2$는 단정도실수에서 24 비트, 배정도실수에서는 53 비트이다. 제곱근의 지수부 연산은 $\sqrt[N]{2^{p \bmod N}}\times 2^{(\frac{p}{N})}$를 계산하는 것이다. $\sqrt[N]{2^{p \bmod N}}$은 상수이므로 $\sqrt[N]{1.f_2}$을 계산한 후에 $\sqrt[N]{2^{p \bmod N}}$을 곱한다. $2^{(\frac{p}{N})}$은 가수부 처리와 별도의 하드웨어에 의해서 병렬적으로 처리하므로 본 논문에서는 생략한다.

부동소수점수 F의 가수부 $1.f_2$는 식 (3)과 같이 두 부분으로 나눌 수 있다.

$$1.f_2=1.g_2+h_2 \tag{3}$$

식 (3)에서 g와 h의 길이를 각각 n_g 및 n_h 비트로 정의한다. h는 '$0\le h<2^{-n_g}$'이고, h의 최대값은 '$h_{\max}=2^{-n_g}-2^{-n_g-n_h}$'이다. 반복식의 수렴 속도를 빠르게 하기 위해서 식 (4)를 근사계산하여 테이블에 미리 작성해놓는다.

$$T=\sqrt[N]{F}-e_0 \tag{4}$$

$$X_0=T^NF^{N-1}=(1-e_0F^{(N-1)/N})^N$$

$$Y_0=TF=\sqrt[N]{F}(1-e_0F^{(N-1)/N})=\sqrt[N]{F}\sqrt[N]{X_0}$$

초기 근사값 X_0와 Y_0는 ROM에 저장하거나 또는 별도의 회로를 사용해서 산출하기도 한다[16]. T는 $\sqrt[N]{1.g}$의 근사계산이므로 $T=\sqrt[N]{F}-e_0$이다. e_0는 근사에 따른 초기 오차이다.

$a_0=\frac{1-X_0}{N}$를 정의하면 $X_0=1-Na_0$가 된다. 이로부터 $\frac{1}{\sqrt[N]{X_0}}$를 테일러급수로 전개하면 식 (5)가 된다. 식 (5)에서 k (k=2, 3, ..)는 차수를 나타낸다.

$$\frac{1}{\sqrt[N]{X_0}}=\frac{1}{\sqrt[N]{1-Na_0}}=\sum_{j=0}^{\infty}c_ja_0^j=$$

$$\sum_{j=0}^{k-1}c_ja_0^j+\sum_{j=k}^{\infty}c_ja_0^j=R_0^{(k)}+e_1 \tag{5}$$

$$\text{where } c_0=c_1=1$$

$$c_m=\frac{\prod_{j=1}^{m-1}(jN+1)}{m!},\ m\ge 2$$

식 (5)의 테일러급수항에서 초기 K항을 취한 것이 $R_i^{(k)}$이고, 나머지 항들이 e_1이다. 이로부터 X_1과 Y_1은 식 (6)과 같이 계산한다.

$$X_1=X_0(R_0^{(k)})^N=(1-e_1\sqrt[N]{X_0})^N \tag{6}$$

$$Y_1=Y_0R_0^{(k)}=\sqrt[N]{F}(1-e_1\sqrt[N]{X_0})=\sqrt[N]{F}\sqrt[N]{X_1}=$$

$$\sqrt[N]{F}-\sqrt[N]{F}\sqrt[N]{X_0}\sum_{j=k}^{\infty}c_j a_0^j=\sqrt[N]{F}-e_{y1}$$

$$e_{y1}=\sqrt[N]{F}\sqrt[N]{X_0}\sum_{j=k}^{\infty}c_j a_0^j \fallingdotseq c_k a_0^k\sqrt[N]{F}$$

식 (6)에서 Y_1은 $\sqrt[N]{F}$의 첫 번째 근사값이고 e_{y1}은 오차값이다. X_i, Y_i로부터 X_{i+1}, Y_{i+1}을 구하는 반복식은 식 (7)과 같이 된다.

$$a_i=\frac{1-X_i}{N} \tag{7}$$

$$R_i^{(k)}=\sum_{j=0}^{k-1}c_j a_i^j$$

$$X_{i+1}=X_i(R_i^{(k)})^N=(1-e_{i+1}\sqrt[N]{X_i})^N$$

$$Y_{i+1}=Y_iR_i^{(k)}=\sqrt[N]{F}(1-e_{i+1}\sqrt[N]{X_i})=$$

$$\sqrt[N]{F}-\sqrt[N]{F}\sqrt[N]{X_i}\sum_{j=k}^{\infty}c_j a_i^j=\sqrt[N]{F}-e_{y(i+1)}$$

$$e_{y(i+1)}\fallingdotseq c_k a_i^k\sqrt[N]{F}$$

식 (7)의 $e_{y(i+1)}$에서 c_k는 상수이고 $\sqrt[N]{F}$의 최대값은 알고 있다. 그러므로 $|a_i|<2^{-t}$이면 $e_{y(i+1)}$의 최대값을 알 수 있다. $|a_i|<2^{-t}$인 조건은 소수점 이하 연속되는 '0' 또는 '1'의 개수가 t보다 많다는 것으로 간단한 하드웨어로 판단이 용이하다. i+1번째 오차 $e_{y(i+1)}$가 일정한 값 이하가 되면 $Y_{i+1}=\sqrt[N]{F}$으로 반복연산을 종료한다.

2.2 K차 골드스미스 제곱근 알고리즘

F의 제곱근 $Y_n=\sqrt{F}$를 계산하기 위한 초기값은 식(8)과 같다.

$$T=\sqrt{F}-e_0 \tag{8}$$

$$X_0=T^2F=(1-e_0\sqrt{F})^2$$

$$Y_0=TF=\sqrt{F}(1-e_0\sqrt{F})=\sqrt{F}\sqrt{X_0}$$

$a_i=\frac{1-X_i}{2}$를 정의하고 $\frac{1}{\sqrt{X_i}}$를 테일러급수로 전개하면 식 (9)가 된다.

$$\frac{1}{\sqrt{X_i}}=1+a_i+\frac{3}{2}a_i^2+\frac{5}{2}a_i^3+\frac{35}{8}a_i^4+\cdots= \tag{9}$$

$$\sum_{j=0}^{k-1}c_j a_i^j+\sum_{j=k}^{\infty}c_j a_i^j=R_i^{(k)}+e_i$$

K=2,3,4에 대한 제곱근의 반복식을 간략하게 정리하면 식 (10)과 같다.

$$a_i=\frac{1-X_i}{2} \tag{10}$$

(K=2) $R_i^{(2)}=1+a_i$

$$X_{i+1}=X_i(R_i^{(2)})^2$$

$$Y_{i+1}=Y_iR_i^{(2)}\fallingdotseq\sqrt{F}-\frac{3\sqrt{F}}{2}a_i^2$$

(K=3) $R_i^{(3)}=1+a_i+\frac{3a_i^2}{2}$

$$X_{i+1}=X_i(R_i^{(3)})^2$$

$$Y_{i+1}=Y_iR_i^{(3)}\fallingdotseq\sqrt{F}-\frac{5\sqrt{F}}{2}a_i^3$$

(K=4) $R_i^{(4)}=1+a_i+\frac{3a_i^2}{2}+\frac{5a_i^3}{3}$

$$X_{i+1}=X_i(R_i^{(4)})^2$$

$$Y_{i+1}=Y_iR_i^{(4)}\fallingdotseq\sqrt{F}-\frac{35\sqrt{F}}{8}a_i^4$$

식 (10)으로부터 $|a_i|<2^{-t}$이면 i+1번째 오차 e_{i+1}은 식 (11)이 된다.

$$e_{i+1}^{(2)}\fallingdotseq\frac{3\sqrt{F}a_i^2}{2}<\frac{3\sqrt{F}}{2}2^{-2t}<2^{-2t+2} \tag{11}$$

$$e_{i+1}^{(3)}\fallingdotseq\frac{5\sqrt{F}a_i^3}{2}<\frac{5\sqrt{F}}{2}2^{-3t}<2^{-3t+2}$$

$$e_{i+1}^{(4)}\fallingdotseq\frac{35\sqrt{F}a_i^4}{8}<\frac{35\sqrt{F}}{8}2^{-4t}<2^{-4t+4}$$

2.3 K차 골드스미스 세제곱근 알고리즘

F의 세제곱근 $Y_n=\sqrt[3]{F}$를 계산하기 위한 초기값은 식(12)와 같다.

$$T=\sqrt[3]{F}-e_0 \tag{12}$$

$$X_0=T^3F^2=(1-e_0F^{2/3})^3$$

$$Y_0=TF=\sqrt[3]{F}(1-e_0F^{2/3})=\sqrt[3]{F}\sqrt[3]{X_0}$$

$a_i=\frac{1-X_i}{3}$를 정의하면 $\frac{1}{\sqrt[3]{X_i}}$를 테일러급수로 전개하면 식 (13)이 된다.

$$\frac{1}{\sqrt[3]{X_i}}=1+a_i+2a_i^2+\frac{14}{3}a_i^3+\frac{91}{3}a_i^4+\cdots= \tag{13}$$

$$\sum_{j=0}^{k-1}c_j a_i^j+\sum_{j=k}^{\infty}c_j a_i^j=R_i^{(k)}+e_i$$

K=2,3,4에 대한 세제곱근의 반복식을 간략하게 정

리하면 식 (14)와 같다..

$$a_i = \frac{1-X_i}{3} \tag{14}$$

(K=2) $R_i^{(2)} = 1 + a_i$

$$X_{i+1} = X_i (R_i^{(2)})^3$$

$$Y_{i+1} = Y_i R_i^{(2)} \fallingdotseq \sqrt[3]{F} - 2a_i^2 \sqrt[3]{F}$$

(K=3) $R_i^{(3)} = 1 + a_i + 2a_i^2$

$$X_{i+1} = X_i (R_i^{(3)})^3$$

$$Y_{i+1} = Y_i R_i^{(3)} \fallingdotseq \sqrt[3]{F} - \frac{14\sqrt[3]{F}}{3} a_i^3$$

(K=4) $R_i^{(4)} = 1 + a_i + 2a_i^2 + \frac{14a_i^3}{3}$

$$X_{i+1} = X_i (R_i^{(4)})^3$$

$$Y_{i+1} = Y_i R_i^{(4)} \fallingdotseq \sqrt[3]{F} - \frac{7\sqrt[3]{F}}{3} a_i^4$$

식 (14)로부터 $|a_i| < 2^{-t}$이면 i+1번째 오차 e_{i+1}은 식 (15)가 된다.

$$e_{i+1}^{(2)} \fallingdotseq \sqrt[3]{F} 2a_i^2 < 2^{-2t+1} \sqrt[3]{F} < 2^{-2t+2} \tag{15}$$

$$e_{i+1}^{(3)} \fallingdotseq \frac{14\sqrt[3]{F} a_i^3}{3} < \frac{14\sqrt[3]{F}}{3} 2^{-3t} < 2^{-3t+4}$$

$$e_{i+1}^{(4)} \fallingdotseq \frac{7\sqrt[3]{F} a_i^4}{3} < \frac{35\sqrt[3]{F}}{3} 2^{-4t} < 2^{-4t+5}$$

3. N차 역제곱근 K차 골드스미트 알고리즘

3.1 N차 역제곱근 K차 골드스미트 알고리즘

부동소수점 수 F의 N차 역제곱근 $Y_n = \frac{1}{\sqrt[N]{F}}$은 초기값 Y_0를 정의하고, 반복식으로 $Y_i (i = 1, \ldots, n)$을 구하기 위해서 식 (16)을 근사계산하여 테이블에 미리 작성해놓는다.

$$T = \frac{1}{\sqrt[N]{F}} - e_0 \tag{16}$$

$$X_0 = T^N F = (1 - e_0 \sqrt[N]{F})^N$$

$$Y_0 = T = \frac{\sqrt[N]{X_0}}{\sqrt[N]{F}}$$

$a_0 = \frac{1-X_0}{N}$를 정의하고 $\frac{1}{\sqrt[N]{X_0}}$를 테일러급수로 전개하면 식 (5)가 된다. 이로부터 X_1과 Y_1은 식 (17)과 같이 계산한다.

$$X_1 = X_0 (R_0^{(k)})^N = (1 - e_1 \sqrt[N]{X_0})^N \tag{17}$$

$$Y_1 = Y_0 R_0^{(k)} = \frac{1 - e_1 \sqrt[N]{X_0}}{\sqrt[N]{F}} = \frac{\sqrt[N]{X_1}}{\sqrt[N]{F}}$$

$$\frac{1}{\sqrt[N]{F}} - \frac{\sqrt[N]{X_0}}{\sqrt[N]{F}} \sum_{j=k}^{\infty} c_j a_0^j = \frac{1}{\sqrt[N]{F}} - e_{y1}$$

$$e_{y1} = \frac{\sqrt[N]{X_0}}{\sqrt[N]{F}} \sum_{j=k}^{\infty} c_j a_0^j \fallingdotseq \frac{c_k a_0^k}{\sqrt[N]{F}}$$

이어서 X_i, Y_i로부터 X_{i+1}, Y_{i+1}을 구하는 반복식은 식 (18)과 같이 된다.

$$a_i = \frac{1-X_i}{N} \tag{18}$$

$$R_i^{(k)} = \sum_{j=0}^{k-1} c_j a_i^j$$

$$X_{i+1} = X_i (R_i^{(k)})^N = (1 - e_{i+1} \sqrt[N]{X_i})^N$$

$$Y_{i+1} = Y_i R_i^{(k)} = \frac{1 - e_{i+1} \sqrt[N]{X_i}}{\sqrt[N]{F}} =$$

$$\frac{1}{\sqrt[N]{F}} - \frac{\sqrt[N]{X_i}}{\sqrt[N]{F}} \sum_{j=k}^{\infty} c_j a_i^j = \frac{1}{\sqrt[N]{F}} - e_{y(i+1)}$$

$$e_{y(i+1)} \fallingdotseq \frac{c_k a_i^k}{\sqrt[N]{F}}$$

식 (18)의 $e_{y(i+1)}$에서 c_k는 상수이고 $\frac{1}{\sqrt[N]{F}}$의 최대값은 알고 있다. 그러므로 $|a_i| < 2^{-t}$이면 $e_{y(i+1)}$의 최대값을 알 수 있다. i+1번째 오차 $e_{y(i+1)}$가 일정한 값 이하가 되면 반복연산을 종료한다.

3.2 K차 골드스미스 역제곱근 알고리즘

F의 역제곱근 $\frac{1}{\sqrt{F}}$를 계산하기 위한 초기값은 식 (19)와 같다.

$$T = \frac{1}{\sqrt{F}} - e_0 \tag{19}$$

$$X_0 = T^2 F = (1 - e_0 \sqrt{F})^2$$

$$Y_0 = T = \frac{1 - e_0 \sqrt{F}}{\sqrt{F}} = \frac{\sqrt{X_0}}{\sqrt{F}}$$

K=2,3,4에 대한 역제곱근의 반복식을 간략하게 정리하면 식 (20)과 같다.

$$a_i = \frac{1-X_i}{2} \tag{20}$$

(K=2) $R_i^{(2)} = 1 + a_i$

$$X_{i+1} = X_i (R_i^{(2)})^2$$

$$Y_{i+1} = Y_i R_i^{(2)} \fallingdotseq \frac{1}{\sqrt{F}} - \frac{3}{2\sqrt{F}} a_i^2$$

(K=3) $R_i^{(3)} = 1 + a_i + \frac{3a_i^2}{2}$

$$X_{i+1} = X_i (R_i^{(3)})^2$$

$$Y_{i+1} = Y_i R_i^{(3)} \fallingdotseq \frac{1}{\sqrt{F}} - \frac{5}{2\sqrt{F}} a_i^3$$

(K=4) $R_i^{(4)} = 1 + a_i + \frac{3a_i^2}{2} + \frac{5a_i^3}{3}$

$$X_{i+1} = X_i (R_i^{(4)})^2$$

$$Y_{i+1} = Y_i R_i^{(4)} \fallingdotseq \frac{1}{\sqrt{F}} - \frac{35}{8\sqrt{F}} a_i^4$$

식 (20)으로부터 $|a_i| < 2^{-t}$이면 i+1번째 오차 e_{i+1}은 식 (21)이 된다.

$$e_{i+1}^{(2)} \fallingdotseq \frac{3a_i^2}{2\sqrt{F}} < \frac{3}{2\sqrt{F}} 2^{-2t} < 2^{-2t+1} \quad (21)$$

$$e_{i+1}^{(3)} \fallingdotseq \frac{5a_i^3}{2\sqrt{F}} < \frac{5}{2\sqrt{F}} 2^{-3t} < 2^{-3t+2}$$

$$e_{i+1}^{(4)} \fallingdotseq \frac{35a_i^4}{8\sqrt{F}} < \frac{35}{8\sqrt{F}} 2^{-4t} < 2^{-4t+2}$$

3.3 정확한 제곱근 연산

식 (20)에서 구한 Y_n은 근사값으로 '$Y_n = \frac{1}{\sqrt{F}} - e_n$, $1 < F < 2$'이다. F의 유효자릿수를 w라고 하면, $e_n < 2^{-w-3}$이 될 때까지 반복연산을 수행한다. IEEE-754[17]로 정의되는 단정도와 배정도 부동소수점수에서 w는 각각 24와 53이다. $0.75 * 2^{-w} \le s < 2^{-w}$를 F에 더한 후에 Y_n을 곱하면 식 (22)가 된다.

$$D = (F+s) Y_n = \sqrt{F} - e_n F + s Y_n = \sqrt{F} + e_d \quad (22)$$
$$= 1.f_1 + 0.f_2 * 2^{-w} + e_d$$

식 (22)에서 '$0 \le (0.f_2 * 2^{-w} + e_d) < 2^{-w+1}$'이므로 식 (23)이 성립한다.

$$D = 1.m_1 + 0.m_2 * 2^{-w} \quad (23)$$
$$(1.m_1)^2 = 1.t_1 + 0.t_2 * 2^{-w}$$

식 (23)은 다음의 4가지 경우로 구분된다.

1. $1.t_1 = F$이고 $0.t_2 = 0$이면 $\sqrt{F} = 1.m_1$이고 스티키(sticky) 비트는 0이다.
2. $1.t_1 = F$이고 $0.t_2 \neq 0$이면 $\sqrt{F} = 1.m_1 - 2^{-w}$이고 스티키 비트는 1이다.
3. $1.t_1 < F$이면 $\sqrt{F} = 1.m1$이고 스티키 비트는 1이다.
4. $1.t_1 > F$이면 $\sqrt{F} = 1.m_1 - 2^{-w}$이고 스티키 비트는 1이다.

3.4 K차 골드스미스 역세제곱근 알고리즘

F의 역세제곱근 $\frac{1}{\sqrt[3]{F}}$를 계산하기 위한 초기값은 식(24)와 같다.

$$T = \frac{1}{\sqrt[3]{F}} - e_0 \quad (24)$$

$$X_0 = T^3 F = (1 - e_0 \sqrt[3]{F})^3$$

$$Y_0 = T = \frac{1 - e_0 \sqrt[3]{F}}{\sqrt[3]{F}} = \frac{\sqrt[3]{X_0}}{\sqrt[3]{F}}$$

K=2,3,4에 대한 역세제곱근의 반복식을 간략하게 정리하면 식 (25)와 같다..

$$a_i = \frac{1 - X_i}{3} \quad (25)$$

(K=2) $R_i^{(2)} = 1 + a_i$

$$X_{i+1} = X_i (R_i^{(2)})^3$$

$$Y_{i+1} = Y_i R_i^{(2)} \fallingdotseq \frac{1}{\sqrt[3]{F}} - \frac{2a_i^2}{\sqrt[3]{F}}$$

(K=3) $R_i^{(3)} = 1 + a_i + 2a_i^2$

$$X_{i+1} = X_i (R_i^{(3)})^3$$

$$Y_{i+1} = Y_i R_i^{(3)} \fallingdotseq \frac{1}{\sqrt[3]{F}} - \frac{14}{3\sqrt[3]{F}} a_i^3$$

(K=4) $R_i^{(4)} = 1 + a_i + 2a_i^2 + \frac{14a_i^3}{3}$

$$X_{i+1} = X_i (R_i^{(4)})^3$$

$$Y_{i+1} = Y_i R_i^{(4)} \fallingdotseq \frac{1}{\sqrt[3]{F}} - \frac{7}{3\sqrt[3]{F}} a_i^4$$

식 (25)로부터 $|a_i| < 2^{-t}$이면 i+1번째 오차 e_{i+1}은 식 (26)이 된다.

$$e_{i+1}^{(2)} \fallingdotseq \frac{2a_i^2}{\sqrt[3]{F}} < \frac{2^{-2t+1}}{\sqrt[3]{F}} < 2^{-2t+1} \quad (26)$$

$$e_{i+1}^{(3)} \fallingdotseq \frac{14a_i^3}{3\sqrt[3]{F}} < \frac{14}{3\sqrt[3]{F}} 2^{-3t} < 2^{-3t+3}$$

$$e_{i+1}^{(4)} \fallingdotseq \frac{7a_i^4}{3\sqrt[3]{F}} < \frac{35}{3\sqrt[3]{F}} 2^{-4t} < 2^{-4t+4}$$

4. 연구 결과 비교

본 논문에서 제안한 알고리즘과 기존의 골드스미트 알고리즘을 비교하기 위하여 난수 발생기를 이용하여 단정도실수와 배정도실수를 각각 10^6개를 생성하여 제곱근을 계산하였다. Table 1과 Table 2에 그 결과를 보인다.

종래 골드스미트 제곱근 알고리즘에서는 최대 오차를 고려해서 반복 횟수를 정했다. 즉, 종래 알고리즘에 의한 단정도실수 제곱근은 '96×6' 테이블을 사용하면 12회의 곱셈을 수행하였다, 본 논문에서 제안한 알고리즘에서는 k=2인 경우에 평균 6.91회의 곱셈, k=3인 경우에는 평균 6.41회의 곱셈으로 제곱근을 계산할 수 있었다.

배정도실수 제곱근 연산에서 '192X7' 테이블을 사용하면 종래 골드스미트 알고리즘에서는 12회의 곱셈을 수행하였다. 본 논문에서 제안한 알고리즘에서는 k=2인 경우에 평균 9.15회의 곱셈, k=3인 경우에는 평균 8.72회의 곱셈으로 제곱근을 계산할 수 있었다.

5. 결 론

부동소수점수 Z의 N차 제곱근(square root) 및 세제곱근(cubic root)의 사용빈도는 높지 않으나 계산시간이 오래 걸리므로 전용 회로를 채택한 시스템이 늘고 있다.

Table 1. Number of multiply of single precision square root

Table size	GoldSchmidt[1]	k=2^2	k=3^2
96×6	12	6.91	6.41
192×7	9	5.96	5.67
384×8	6	5.91	5.61

*1 : GoldSchmidt algorithm
*2 : Proposed algorithm

Table 2. Number of multiply of double precision square root

Table size	GoldSchmidt	k=2	k=3
96×6	12	10.38	9.93
192×7	12	9.15	8.72
384×8	9	8.95	8.51

본 논문에서는 골드스미트 제곱근 알고리즘에 테일러급수를 적용하여 가칭 N차 제곱근 K차 골드스미트 알고리즘을 제안하였다. 제안한 알고리즘은 $\sqrt[N]{Z} = \sqrt[N]{F}\sqrt[N]{2^{p \bmod N}} \times 2^{(\frac{p}{N})}$에서 $\sqrt[N]{F}$과 $\frac{1}{\sqrt[N]{F}}$을 반복 곱셈으로 계산한다.

N차 제곱근 $Y_{i+1} = \sqrt[N]{F}$을 연산하기 위하여 초기값 X_0와 Y_0를 다음과 같이 미리 연산하여 테이블 등에 저장해놓는다

$$T = \sqrt[N]{F} - e_0,\ X_0 = T^N F^{N-1},\ Y_0 = TF$$

반복식은 다음과 같다.

$$a_i = \frac{1 - X_i}{N}$$

$$\frac{1}{\sqrt[N]{X_i}} = \sum_{j=0}^{k-1} c_j a_i^j + \sum_{j=k}^{\infty} c_j a_i^j = R_i^{(k)} + e_r$$

$$X_{i+1} = X_i (R_i^{(k)})^N$$

$$Y_{i+1} = Y_i R_i^{(k)} + e_{i+1}$$

N차 역제곱근 $Y_{i+1} = \frac{1}{\sqrt[N]{F}}$을 연산하기 위한 초기값 X_0와 Y_0는 $T = \frac{1}{\sqrt[N]{F}} - e_0,\ X_0 = T^N F,\ Y_0 = T$이고, 반복식은 동일하다.

본 논문에서 제안한 알고리즘은 곱셈만을 사용하므로 하드웨어 구현이 용이하다. 또한 e_{i+1}을 예측할 수 있으므로 유효자릿수까지만 연산할 수 있으므로 불필요한 연산을 줄일 수 있다.

REFERENCE

[1] S. Borkar and A.A. Chien, "The Future of Microprocessors," *Communications ACM*, Vol. 54, No. 5, pp. 67-77, 2011.

[2] I.H. Song, H.J. Kwon, T.K. Kim, and S.H. Lee, "3D Image Representation Using Color Correction Matrix According to the CCT of a Display," *Journal of Korea Multimedia Society*, Vol. 22, No. 1, pp. 55-61, 2019.

[3] P. Montuschi, J.D. Bruguera, L. Ciminiera, and J.A. Pibeiro, "A Digit-by-Digit Algorithm for

mth Root Extraction," *IEEE Transactions on Computers*, Vol. 56, No. 12, pp. 1696-1708, 2007.

[4] Y. Li and W. Chu, "On the Improved Implementations and Performance Evaluation of Digit-by-Digit Integer Restoring and Non-restoring Cube Root Algorithms," *Proceeding of International Conference on Computer, Information and Telecommunication Systems*, pp. 1-5, 2016.

[5] Y. Luo, Y. Wang, H. Sun, and Z. Wang, "CORDIC-Based Architecture for Computing Nth Root and Its Implementation," *IEEE Transactions on Circuits and Systems I*, Vol. 65, Issue 12, pp. 4183-4195, 2018.

[6] C.S. Yan, W.D. Hui, and H.C. Huan, "Design and Implementation of a 64/32-bit Floating-point Division, Reciprocal, Square root, and Inverse Square root Unit," *Proceedings of Solid-State and Integrated Circuits Technology 8th International Conference on*, pp. 1976-1979, 2006.

[7] S.G. Chen and P.Y. Hsieh, "Fast Computation of the Nth Root," *Computers and Mathematics With Applications*, Vol. 17, No. 10, pp. 1423-1427, 1989.

[8] F. Dubeau, "Nth Root Extraction: Double Iteration Process and Newtons's Method," *Journal of Computational and Applied Mathematics*, Vol. 91, Issue 2, pp. 191-198, 1998.

[9] F. Dubeau, "Newton's Method and High-order Algorithm for the Nth Root Computation," *Journal of Computational and Applied Mathematics*, Vol. 224, Issue 1, pp. 66-76, 2009.

[10] J.M. Gutierrez, M.A. Hernandez, and M.A. Salanova, "Calculus of Nth Roots and Third Order Iterative Methods," *Nonlinear Analysis* Vol. 47(4), pp. 2875-2880, 2001.

[11] P.D. Proinov and S.I. Ivanov, "On the Convergence of Halley's Method for Multiple Polynomials Zeros," *Mediterranean Journal of Mathematics*, Vol. 12, No. 2, pp. 555-572, 2015.

[12] S.F. Oberman and M.J. Flynn, "Design Issues in Division and Other Floating Point Operations," *IEEE Transactions on Computer*, Vol. 46, Issue 2, pp. 154-161, 1997.

[13] S.G. Kim and G.Y. Cho, "A Variable Latency Newton-Rapson's Floating Point Number Reciprocal Square Root Computation," *Korea Information Processing Society*, Vol. 12, No. 2, pp. 413-420, 2005.

[14] S.G. Kim, H.B. Song, and G.Y. Cho, "A Variable Latency Goldschmidt's Floating Point Number Square Root Computation," *Korea Institute of Maritime Information and Communication Sciences*, Vol. 9, No. 1, pp. 188-198, 2004.

[15] G.Y. Cho, "Error Corrected K'th Order Goldschmidt's Floating Point Number Division," *Korea Institute of Information and Communication Sciences*, Vol. 19, No. 10, pp. 2341-2349, 2015.

[16] D.D. Sarma and D. Matula, "Measuring and Accuracy of ROM Reciprocal Tables," *IEEE Transactions on Computer*, Vol. 43, No. 8, pp. 932-930, 1994.

[17] IEEE, IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Standard, Std. 754-1985.

조 경 연

1990년 인하대학교 전자공학과 박사
1991년 부산수산대학교 전자계산학과 전임강사
현재 부경대학교 IT융합응용공학과 교수

관심분야 : 컴퓨터구조, 디지털회로설계, 정보보호