ハイパフォーマンス
コンピューティング 55−10
（１９９５． ３． ９）

除算と開平のための積和演算を用いた初期近似手法

伊藤 雅之† 高木 直史‡ 矢島 脩三†

†京都大学 工学部 情報工学科
〒606-01 京都市左京区吉田本町

‡名古屋大学工学部情報工学科
〒464-01 名古屋市千種区不老町

高速乗算器を用いた収束法による除算・開平のための効率のよい初期近似手法を提案する。積和演算を1サイクルで実行できる機能ユニット上での演算を考える。近似には一次関数を用いるが、従来の一次近似と異なり、オペランドの一部のビットを反転した値から近似値を計算する。一次関数の二つの係数は参照テーブルに登録する。従来法と比べて、高精度の近似値を比較的小さなサイズのテーブルを用いて生成できる。本手法の近似により、倍精度(53ビット)の除算・開平が二次の収束アルゴリズムを1回計算するだけで実現でき、単精度(24ビット)では、本手法の近似だけで必要な精度を得ることができる。

Efficient Initial Approximation Methods for Division and Square Root Using a Multiply-Add Unit

Masayuki ITO†, Naofumi TAKAGI‡ and Shuzo YAJIMA†

†Department of Information Science
Faculty of Engineering
Kyoto University, Kyoto 606-01, Japan

‡Department of Information Engineering
Faculty of Engineering
Nagoya University, Nagoya 464-01, Japan

Efficient initial approximation methods for multiplicative division and square root are proposed. A functional unit where one multiply-accumulate operation can be executed in one cycle is assumed. An approximation is calculated from a linear function of a modified operand. The modification is just some bit-inversions. Two coefficients of the linear function are read through table look-up. High-precision approximations can be produced through comparatively small look-up tables. Our initial approximation methods make only one calculation of converging algorithms yield double precision (53-bits) quotients and square roots. They directly produce single precision (24-bits) quotients and square roots.

1 Introduction

With the increasing availability of high-speed multiplication units, convergence-type algorithms have become advantageous to the fast calculation of division and square root. Generally, converging methods adopt an initial approximation to the desired value and improve it by a converging algorithm, e.g., Newton-Raphson's or Goldschmidt's. Efficient initial approximations reduce the number of iterations of converging algorithms and achieve high-speed division and square root.

We deal with division and square root operations on the mantissa of floating point numbers. We focus on generating initial approximations. We assume a functional unit on which one multiply-accumulate operation on three fractional numbers $(A \times B) + C$ can be executed in one cycle. Many functions can be calculated efficiently on such a multiply-add unit. We propose efficient initial approximation methods suitable for such a multiply-add unit.

For the generation of initial approximations, look-up tables are commonly used. As the development of the VLSI technology, large size of look-up tables can be practically used, which enables fast division and square root by starting with high-precision approximations.

The simplest way is directly reading an initial approximation to the desired value through table look-up using some most significant digits of an operand as an index [1]. Beside this direct approximation method, a linear approximation method can be employed. In this case the table keeps the two coefficients of the linear function. The calculation of the linear function requires one multiply-accumulate operation, and the result is roughly twice as many bits of accuracy as that achieved by the direct approximation method. As initial approximations for division and square root, linear approximation paying the penalty of one cycle seems to be most efficient among polynomial approximations.

We propose new initial approximation methods for division and square root. Each method is a modification of the linear approximation and requires one multiply-accumulate operation. High-precision approximations can be produced through comparatively small look-up tables. Our methods make only one calculation of converging algorithms yield double precision (53-bits) quotients and square roots. They directly produce single precision (24-bits) quotients and square roots. They are effective for multiplicative algorithms such as Newton-Raphson's and Goldschmidt's.

This report is organized as follows. In Section 2, we review some general initial approximation methods. In Section 3, we propose new initial approximation methods. In Section 4, we compare our methods to the conventional methods from the viewpoint of precision and required size of look-up tables. A conclusion is shown in Section 5.

2 Conventional Methods

In this section, we review some conventional initial approximation methods for division and square root. The simplest way is directly reading an initial approximation to the desired value through table look-up. Linear approximation by a linear function whose two coefficients are kept in a look-up table can also be adopted.

Polynomial approximation by a polynomial of degree $n (\geq 2)$ may be employed [2]. It needs n multiply-accumulate operations and the look-up table must keep $(n + 1)$ coefficients of the polynomial. It is inefficient as an initial approximation method, because errors decrease faster by adopting an iterative higher order converging algorithm such as the Newton-Raphson method. For this reason, here we only discuss direct approximation (DA in short) and linear approximation (LA in short).

A. Reciprocal

Multiplicative division begins with an initial approximation R_0 to the reciprocal of a given divisor $Y = [1.y_1y_2 \cdots y_my_{m+1} \cdots]$, where $1 \leq Y < 2$. R_0 then has the form $[0.1r_1r_2 \cdots r_t]$. After R_0 is formed, an iterative converging algorithm is adopted to achieve the desired precision.

In the DA method, when the table is addressed with the m most significant bits of the divisor Y, the look-up table keeps 2^m approximations for each of 2^m subintervals of Y. The size of the table is $2^m \times t$ bits. To give the best value for the subinterval $[p, p+2^{-m})$, where $p = [1.y_1y_2 \cdots y_m]$, the look-up table should contain the value $\frac{1}{2}(\frac{1}{p} + \frac{1}{p+2^{-m}})$. The worst error occurs on the first interval $p = 1$. The total error of the DA method ϵ_{DA} considering the error due to storing only t bits of R_0 in the table is

$$|\epsilon_{DA}| = |R_0 - \frac{1}{Y}| < 2^{-m-1} + 2^{-t-2}. \quad (1)$$

We can select any combination of m and t. Here, in order to make the error term caused by storing t bits in the table smaller than that caused by using m bits to address the table, we adopt a reasonable combination $t = m$. Then ϵ_{DA} becomes

$$|\epsilon_{DA}| < \frac{3}{2} \cdot 2^{-m-1} \quad (2)$$

and the table is of size $2^m \times m$ bits.

The LA method adopts a linear function $(-C_1 \cdot Y + C_0)$ for an initial approximation R_0. These two coefficients C_1 and C_0 are read through table look-up addressed with the m most significant bits of Y and have the length of t bits for each. It corresponds to decide the most appropriate linear functions for each of 2^m subintervals $[p\ ,\ p+2^{-m})$. Therefore, the table is of size $2^m \times t \times 2$.

Now we decide C_1 and C_0. Let $E(Y)$ be the error function:

$$E(Y) = -C_1 \cdot Y + C_0 - \frac{1}{Y}. \quad (3)$$

Differentiating (3) yields $E'(Y) = \frac{1}{Y^2} - C_1$. Hence,

$$E(Y)_{max} = E(\frac{1}{\sqrt{C_1}}) = C_0 - 2\sqrt{C_1}. \quad (4)$$

To minimize $|E(Y)|_{max}$, errors of both endpoints of each subinterval, i.e., $E(p)$ and $E(p+2^{-m})$, should have the same value and it should also be equal to $-E(Y)_{max}$. From these conditions we get

$$\begin{cases} C_1 &= \frac{1}{p\cdot(p+2^{-m})}, \\ C_0 &= \frac{p+2^{-m-1}+\sqrt{p\cdot(p+2^{-m})}}{p\cdot(p+2^{-m})}, \end{cases} \quad \text{and} \quad (5)$$

$$|E(Y)|_{max} < \frac{1}{p^3} \cdot 2^{-2m-3}. \quad (6)$$

The total error of the linear approximation ϵ_{LA} considering truncation errors due to storing only t bits of the two coefficients in the table is

$$|\epsilon_{LA}| < \frac{1}{p^3} \cdot 2^{-2m-3} + p \cdot 2^{-t-1} + 2^{-t-1}. \quad (7)$$

Substituting the worst case $p = 1$ and setting a reasonable combination $t = 2m + 3$ yields

$$|\epsilon_{LA}| < 2^{-2m-2} \quad (8)$$

and the table is of size $(2m+3) \times 2^m \times 2$.

B. Square Root Reciprocal

Among many multiplicative square root methods, the Newton-Raphson method for calculating $1/\sqrt{X}$ is commonly used due to its quadratic convergence [3]. It begins with an initial approximation S_0 to the square root reciprocal of a given operand $X = [1.x_1x_2 \cdots x_mx_{m+1} \cdots]$. After iterations of the converging formula to achieve the desired precision, it is multiplied by X to obtain $\sqrt{X}$.

In the case of the DA method for square root reciprocal, we use an m-bits-in, t-bits-out look-up table for producing S_0 as the case of reciprocal. For square root, however, m bits index for the table consists of the $(m-1)$ most significant bits of the mantissa X, i.e., $[x_1x_2x_3 \cdots x_{m-1}]$, and the last bit of the exponent part of the original floating point number [3]. This is because the square root of X and $X/2$ differ by a factor of $\sqrt{2}$ unlike reciprocal. In other words, initial approximation S_0 depends on whether the exponent part of an original floating point number is even or odd. S_0 has the form $[0.1s_1s_2\cdots s_t]$ satisfying $1/2 < S_0 \leq 1$. To give the best value for the subinterval $[u, u + 2^{-m+1})$, where $u = [1.x_1x_2\cdots x_{m-1}]$, the look-up table should contain the value $\frac{1}{2}(1/\sqrt{u} + 1/\sqrt{u + 2^{-m+1}})$. The worst error occurs on the first subinterval $u = 1$. The total error of this DA method δ_{DA} considering the error due to storing only t bits in the table is

$$|\delta_{DA}| < 2^{-m-1} + 2^{-t-2}. \tag{9}$$

Setting a reasonable combination $t = m$ yields

$$|\delta_{DA}| < \frac{3}{2}\cdot 2^{-m-1} \tag{10}$$

and the table is of size $2^m \times m$ bits.

The LA method for square root reciprocal adopts a linear function $(-D_1 \cdot X + D_0)$. These two coefficients D_1 and D_0 are read through table look-up addressed with the $(m - 1)$ most significant bits of X together with the last bit of the exponent, and have the length of t bits for each. Adopting the same scheme in the case of reciprocal, we decide D_1 and D_0 and get the total error of the LA method δ_{LA}:

$$\begin{cases} D_1 = \frac{1}{\sqrt{u\cdot(u+2^{-m+1})}(\sqrt{u}+\sqrt{u+2^{-m+1}})}, \\ D_0 = \frac{1}{2\sqrt{u}} + D_1\cdot\frac{u}{2} + \frac{3}{4}(2D_1)^{\frac{1}{3}} \quad \text{and} \end{cases} \tag{11}$$

$$|\delta_{LA}| < \frac{3}{4u^2\sqrt{u}}\cdot 2^{-2m-2} + u\cdot 2^{-t-1} + 2^{-t-1}. \tag{12}$$

Substituting the worst case $u = 1$ and setting $t = 2m + 2$ yields:

$$|\delta_{LA}| < 2^{-2m-1} \tag{13}$$

and the table is of size $2^m \times (2m + 2) \times 2$.

3 New Initial Approximation Methods

In this section we propose new initial approximation methods. Each method is a modification of the linear approximation and we call it ML method in short. It requires one multiply-accumulate operation and can be executed in one cycle on a multiply-add unit. High-precision approximations can be produced through comparatively small look-up tables. Our methods make only one calculation of converging algorithms yield double precision (53-bits) quotients and square roots. They directly produce single precision (24-bits) quotients and square roots.

A. Reciprocal

In the case of the LA method for the reciprocal of a given divisor $Y = [1.y_1y_2\cdots y_my_{m+1}\cdots]$, a linear function $(-C_1\cdot Y + C_0)$ has been adopted. Here we adopt a modified linear function $[A_1 \cdot (2p + 2^{-m} - Y) + A_0]$ for the approximation to $1/Y$. Recall that $p = [1.y_1y_2\cdots y_m]$. Since $C_0 \simeq C_1 \cdot (2p + 2^{-m})$ in the LA method, $A_1 \cdot (2p + 2^{-m} - Y)$ can form almost the same value as $(-C_1\cdot Y + C_0)$ by setting $A_1 \simeq C_1$. We can use A_0 to improve the approximation.

We form $(2p + 2^{-m} - Y)$ as follows. Let

$$\begin{aligned} q &= [0.0\ 0\ \cdots 0\ y_{m+1}y_{m+2}y_{m+3}\cdots], \\ p' &= [1.y_1y_2\cdots y_m1], \quad \text{and} \\ q' &= [0.0\ 0\ \cdots 0\ \tilde{\bar{y}}_{m+1}y_{m+2}y_{m+3}\cdots], \end{aligned}$$

where $\tilde{\bar{0}} = -1$ and $\tilde{\bar{1}} = 0$. Then $Y = p + q = p' + q'$ and $|q'| < 2^{-m-1}$. We get the relation

$$\begin{aligned} &2p + 2^{-m} - Y = p' - q' \\ &= [1.y_1y_2\cdots y_m\tilde{y}_{m+1}\tilde{y}_{m+2}\tilde{y}_{m+3}\cdots], \end{aligned}$$

where $\tilde{1} = 0$ and $\tilde{0} = 1$. Thus $(2p + 2^{-m} - Y)$ can be obtained only by inverting less significant bits than y_m in Y.

To determine A_1 and A_0, we calculate $[1/Y - C_1\cdot(2p + 2^{-m} - Y)]$.

$$\begin{aligned} &\frac{1}{Y} - C_1\cdot(2p + 2^{-m} - Y) \\ = &\ \frac{1}{p+q} - \frac{(p + 2^{-m}) - q)}{p\cdot(p + 2^{-m})} \\ = &\ \frac{1}{p}\cdot(1 + \frac{q}{p})^{-1} - [\frac{1}{p} - \frac{q}{p\cdot(p + 2^{-m})}] \\ = &\ (\frac{1}{p} - \frac{q}{p^2} + \frac{q^2}{p^3}\cdots) - [\frac{1}{p} - \frac{q}{p^2}\cdot(1 + \frac{2^{-m}}{p})^{-1}] \\ = &\ \frac{q^2 - q\cdot 2^{-m}}{p^3} + o(2^{-3m}) \end{aligned}$$

$$= -\frac{2^{-2m-2}}{p^3} + \frac{q'^2}{p^3} + o(2^{-3m}). \tag{14}$$

(14) indicates that the approximation error can become smaller than 2^{-3m} by setting

$$\begin{cases} A_1 = C_1 - \frac{2^{-2m-2}}{p^4} & \text{and} \\ A_0 = \frac{q'^2}{p^3} \end{cases} \tag{15}$$

We use an m-bits-in table for A_1. As A_0 depends on both p and q', the table for A_0 should be addressed with the m_p most significant bits of p and the m_q most significant bits of q. Let $\delta_p(|\delta_p| < 2^{-m_p-1})$ and $\delta_q(|\delta_q| < 2^{-m-m_q-1})$ be errors coming from using only m_p and m_q bits for the index the table respectively. Then,

$$|A_0 - \tfrac{q'^2}{p^3}| = |\tfrac{(q'+\delta_q)^2}{(p+\delta_p)^3} - \tfrac{q'^2}{p^3}| \\ < (3\cdot 2^{-m_p-1} + 2\cdot 2^{-m_q} + 2^{-2m_q})\cdot 2^{-2m-2}. \tag{16}$$

Suppose the table width for A_1 and A_0 be t_1 and t_0 respectively. From (16) and $|q'| < 2^{-m-1}$, the total error of the modified linear approximation ϵ_{ML} becomes

$$|\epsilon_{ML}| < (3\cdot 2^{-m_p-1} + 2\cdot 2^{-m_q} + 2^{-2m_q} + \\ 2^{-t_0-1}) \times 2^{-2m-2} + 2^{-t_1-1} + o(2^{-3m}). \tag{17}$$

In order to make the total size of two look-up tables nearly even to the case of the LA method, we set $m_p = \lfloor\frac{m}{2}\rfloor$, $m_q = \lceil\frac{m}{2}\rceil$, $t_0 = \lceil\frac{m}{2}\rceil + 1$, and $t_1 = \lfloor\frac{5m}{2}\rfloor + 4$. Then,

$$|\epsilon_{ML}| < 2^{-2.5m} \tag{18}$$

and the total table size is $2^m \times (3m+5)$ bits.

This result shows that the ML method is more effective than the LA method when $m > 4$. For instance, when $m = 10$, the ML method produces 25 correct bits using 35K bits table, which is three bits better approximation using 24% smaller table than that by the LA method.

B. Square Root Reciprocal

In the case of square root reciprocal, we adopt a modified linear function $[B_1 \cdot \frac{1}{2}(3u + 3\cdot 2^{-m} - X) + B_0]$ for an approximation to $1/\sqrt{X}$, where $X = [1.x_1x_2\cdots x_{m-1}x_m\cdots]$. Recall that $u = [1.x_1x_2\cdots x_{m-1}]$. Since $D_0 \simeq D_1\cdot(3u + 3\cdot 2^{-m})$ in the LA method, $B_1\cdot(\frac{3}{2}u + \frac{3}{2}\cdot 2^{-m} - \frac{X}{2})$ can form almost the same value as $(-D_1\cdot X + D_0)$ by setting $B_1 \simeq 2D_1$. We can use B_0 to improve the approximation.

We form $(\frac{3}{2}u + \frac{3}{2}\cdot 2^{-m} - \frac{X}{2})$ as follows. Let

$$\begin{aligned} v &= [0.\,0\ 0\cdots 0\ x_m x_{m+1} x_{m+2}\cdots], \\ u' &= [1.x_1x_2\cdots x_{m-1}1], \quad \text{and} \\ v' &= [0.\,0\ 0\cdots 0\ \tilde{\bar{x}}_m x_{m+1} x_{m+2}\cdots], \end{aligned}$$

where $\tilde{\bar{0}} = -1$ and $\tilde{\bar{1}} = 0$. Then $X = u + v = u' + v'$ and $|v'| < 2^{-m}$. We get the relation

$$\frac{3}{2}u + \frac{3}{2}\cdot 2^{-m} - \frac{X}{2} = u' - \frac{v'}{2} \\ = [1.x_1x_2\cdots x_{m-1}\tilde{x}_m x_m \tilde{x}_{m+1}\tilde{x}_{m+2}\cdots],$$

where $\tilde{1} = 0$ and $\tilde{0} = 1$. Thus we need only small additional hardware to obtain $(\frac{3}{2}u + \frac{3}{2}\cdot 2^{-m} - \frac{X}{2})$.

To determine B_1 and B_0, we calculate $[1/\sqrt{X} - D_1\cdot(\frac{3}{2}u + \frac{3}{2}\cdot 2^{-m} - \frac{X}{2})]$.

$$\begin{aligned} &\frac{1}{\sqrt{X}} - 2D_1\cdot(\frac{3}{2}u + \frac{3}{2}\cdot 2^{-m} - \frac{X}{2}) \\ = &\ \frac{1}{u+v} - \frac{(3u - X) + 3\cdot 2^{-m}}{\sqrt{u\cdot(u + 2^{-m+1})}(\sqrt{u} + \sqrt{u + 2^{-m+1}})} \\ = &\ (\frac{1}{\sqrt{u}} - \frac{v}{2u\sqrt{u}} + \frac{3v^2}{8u^2\sqrt{u}} - \cdots) - (2u - v + \\ &3\cdot 2^{-m}) \times (\frac{1}{2u\sqrt{u}} - \frac{3\cdot 2^{-m+1}}{8u^2\sqrt{u}} + \frac{5\cdot 2^{-2m+2}}{16u^3\sqrt{u}} - \cdots) \\ = &\ \frac{3v^2 - 3v\cdot 2^{-m+1} - 2^{-2m+1}}{8u^2\sqrt{u}} + o(2^{-3m+3}) \\ = &\ \frac{5\cdot 2^{-2m}}{8u^2\sqrt{u}} + \frac{3v'^2}{8u^2\sqrt{u}} + o(2^{-3m+3}). \end{aligned} \tag{19}$$

(19) indicates that the approximation error can become smaller than 2^{-3m+3} by setting

$$\begin{cases} B_1 = 2D_1 + \frac{5\cdot 2^{-2m}}{8u^3\sqrt{u}} & \text{and} \\ B_0 = \frac{3v'^2}{8u^2\sqrt{u}} \end{cases} \tag{20}$$

We use an m-bits-in table for B_1. The m bits consist of the $(m-1)$ most significant bits of X and the last bit of the exponent. As B_0 depends on both u and v', the table for B_0 should be addressed with the $(m_u - 1)$ most significant bits of u and the m_v most significant bits of v together with the last bit of the exponent. Let t_1 and t_0 be the table width for B_1 and B_0 respectively. Then the total error of the ML method for square root reciprocal δ_{ML} becomes

$$|\delta_{ML}| < \frac{3}{8}\cdot(\frac{5}{2}\cdot 2^{-m_u} + 2\cdot 2^{-m_v} + 2^{-t_0-1})\times 2^{-2m} + 2^{-t_1-1} + o(2^{-3m+3}). \quad (21)$$

Setting $m_u = \lfloor\frac{m}{2}\rfloor, m_v = \lceil\frac{m}{2}\rceil, t_0 = \lceil\frac{m}{2}\rceil$, and $t_1 = \lfloor\frac{5m}{2}\rfloor + 3$ results in:

$$|\delta_{ML}| < 2^{-2.5m+1} \quad (22)$$

and the total table size is $2^m \times (3m+3)$ bits. This result is very similar to the case of reciprocal and shows that the ML method is more effective than the LA method when $m > 4$. For instance, when $m = 10$, the ML method gives three bits better approximation using 25% smaller look-up tables than the LA method.

C. Square Root

The ML method is so efficient that the generated approximation may already achieve the desired precision. In such a case, the ML method directly for $\sqrt{X}$ should be adopted and no converging algorithm is needed. It forms $\sqrt{X}$ by one multiply-accumulate operation, if look-up tables of enough size can be used for the desired accuracy.

We first consider the LA method directly for $\sqrt{X}$ in preparation for the ML method. It adopts a linear function $(E_1 \cdot X + E_0)$. These two coefficients E_1, E_0 and the total error of the LA method ξ_{LA} is

$$\begin{cases} E_1 = \frac{1}{\sqrt{u}+\sqrt{u+2^{-m+1}}}, \\ E_0 = \frac{2u+2^{-m+1}+6\sqrt{u}\sqrt{u+2^{-m+1}}}{4(\sqrt{u}+\sqrt{u+2^{-m+1}})}, \quad \text{and} \end{cases} \quad (23)$$

$$|\xi_{LA}| < 2^{-2m-4} + u\cdot 2^{-t-1} + 2^{-t-1}. \quad (24)$$

The ML method for $\sqrt{X}$ adopts a modified linear function $[F_1 \cdot \frac{1}{2}(u + 2^{-m} + X) + F_0]$, where

$$\begin{cases} F_1 = 2E_1 - \frac{3\cdot 2^{-2m-4}}{u^2\sqrt{u}} \quad \text{and} \\ F_0 = -\frac{v'^2}{8u\sqrt{u}}. \end{cases} \quad (25)$$

We can form $\frac{1}{2}\cdot(u + 2^{-m} + X)$ as follows:

$$\frac{1}{2}(u+2^{-m}+X) = [1.x_1x_2\cdots x_{m-1}\tilde{x}_m x_m x_{m+1}\cdots]. \quad (26)$$

Using look-up tables of the same size for the ML method for $1/\sqrt{X}$, the total error of the ML method for square root ξ_{ML} becomes

$$|\xi_{ML}| < 2^{-2.5m+1} \quad (27)$$

and the total table size is $2^m \times (3m + 3)$.

4 Comparisons

In this section, we compare the ML method to the DA and the LA method. We adopt the Newton-Raphson method as converging algorithms after each initial approximation. We show how much precision can be obtained with equal numbers of iterations n on a multiply-add unit (MAU in short). We discuss how many cycles and how large look-up tables are required when we calculate single precision and double precision quotients and square roots for the IEEE standard.

The Newton-Raphson method for reciprocal adopts R_0 as an initial approximation and then refine it by the following iterative formula:

$$R_i = (2 - R_{i-1}\cdot Y)\cdot R_{i-1}. \quad (28)$$

R_i converges to $1/Y$ quadratically [3]. After n iterations to achieve the desired accuracy, R_n is multiplied by a dividend Z, where $1 \leq Z < 2$ to obtain a quotient. Calculations of (28) require two cycles on the MAU at each iteration. Thus, n iterations need $2n$ cycles. When we adopt the LA or the ML method, $(2n+1)$ cycles are required to form R_n . Table 1 shows the result. The precision is evaluated by $\lfloor -\log_2|R_n - 1/Y|\rfloor$, which shows the number of correct bits of R_n.

When we calculate double precision (53-bits) quotients, R_n must keep 54 correct bits. From Table 1, in order to achieve the precision of 54 bits with two iterations, m must be 14 for DA, and 6 for LA and ML. Corresponding required table sizes are 224K, 1.9K, and 1.5K bits respectively. If we reduce the number of iterations into one for high-speed division using larger look-up tables, m will be 27, 13, and 11 for DA, LA, and ML respectively. In this case, the corresponding required table sizes are 3400M, 464K, and 76K bits. By the ML method, we can reduce the required table size into about 3/4 to 1/6 of that by the LA method and less than 1/150 of that by the DA method.

For single precision (24-bits) quotients, R_n must keep 25 correct bits. From Table 1, the ML method requires only one cycle without adopting the converging algorithm, if a table of size 35K bits ($m = 10$) is available, while the LA method needs 216K bits ($m = 12$). When we use the converging algorithm once after the DA method with two cycles, the required table size is 104K bits ($m = 13$), which is still much larger than that by the ML method alone with one cycle.

In the case of square root, we compare the ML method to the DA and the LA method, all of which are followed by the multiplicative Newton-Raphson method (NR in short). Its iterative formula is $S_i = \frac{S_{i-1}}{2} \cdot (3 - X \cdot S_{i-1}^2)$ and S_i converges to $1/\sqrt{X}$ quadratically. Each iteration can be executed in three cycles on the MAU. In the final (n-th) iteration, we should calculate $S_n \cdot X$, which can also be done in three cycle. Thus, n iterations of the formula involve $3n$ multiplications. When we adopt the LA or the ML method, $(3n + 1)$ cycles are required on the MAU. Table 2 shows the result. The precision can be evaluated by $\lfloor -\log_2 |S_n \cdot X - \sqrt{X}| \rfloor$, which shows the number of correct bits of $(S_n \cdot X)$.

When we calculate double precision square roots, the result must keep 53 correct bits. From Table 2, if we set the number of iterations $n = 2$, the ML method requires 1.3K bits table ($m = 6$), while LA needs 4.0K bits($m = 7$) and DA needs 224K bits ($m = 14$). If we reduce the number of iterations into one, the ML method requires 156K bits ($m = 12$), while LA needs 448K bits ($m = 13$) and DA needs 3400M bits ($m = 27$). By the ML method, we can reduce the required table sizes into about 1/3 of that by the LA method. The DA method is far inferior to other two methods. It needs one calculation of the converging algorithm to achieve the same accuracy as that given by the LA method alone, which is still worse than that by the ML method. It is because the first iteration of the Newton-Raphson method consuming three cycles is no better than one multiply-accumulate operation of the initial approximation requiring only one cycle.

For single precision square roots, the ML method directly for $\sqrt{X}$ achieves the desired accuracy with only one cycle, if a look-up table of size 33K bits ($m = 10$) is available.

5 Conclusion

We have proposed new initial approximation methods for multiplicative division and square root. Each of our methods is a modification of the conventional linear approximation and requires one multiply-accumulate operation. Since converging algorithms for division and square root use multiplication as a basic operation, the functional unit containing a high-speed multiplier can also be utilized for producing initial approximation.

Our initial approximation methods reduce the size of look-up tables drastically. One multiply-accumulate operation for initial approximations is worth the first iteration of the Newton-Raphson method consuming a few cycles (two for division and three for square root). Our methods can produce even better approximations than the direct approximation followed by the first iteration of the converging algorithms.

For the calculation of double precision quotients and square roots, our method requires 3/4 to 1/6 the size of look-up tables in comparison to the conventional linear approximation. Consequently, the necessary number of iterations of the converging algorithm becomes only one if

76K bits and 156K bits table for division and square root are available respectively.

In the case of single precision, our approximation can directly achieve the desired precision by one multiply-accumulate operation. Look-up tables of practical size 35K bits and 33K bits are required for division and square root respectively.

Acknowledgements

The authors would like to thank the members of Professor Yajima's Laboratory at Kyoto University for their valuable comments on this work.

References

[1] Debjit DasSarma and David W. Matula, Measuring the accuracy of ROM reciprocal tables, IEEE Trans. on Computers, Vol.43, No.8(Aug.1994),932-940

[2] Ned Anderson, Minimum relative error approximations for 1/t, Numerische Mathematic 54 (1988), 117-124

[3] Peter Soderquist and Miram Leeser, Area and Performance Tradeoffs in Floating-Point Division and Square Root Implementations, Technical Report EE-CEG-94-5

Table 1 : The number of correct bits of reciprocals with n iterations using an m-bits-in table

Method	n=0	n=1	n=2	n=3	Table size
DA	—	2m	4m+1	8m+2	$m \times 2^m$
LA	2m+2	4m+4	8m+8	16m+16	$(4m+6) \times 2^m$
ML	2.5m	5m	10m	20m	$(3m+5) \times 2^m$

Table 2: The number of correct bits of square roots with n iterations using an m-bits-in table

Method	n=0	n=1	n=2	n=3	Table size
DA+NR	—	2m	4m−1	8m−1	$m \times 2^m$
LA+NR	—	4m+1	8m+3	16m+5	$(4m+4) \times 2^m$
ML+NR	—	5m−3	10m−6	20m−13	$(3m+3) \times 2^m$
ML for $\sqrt{X}$	2.5m−1	—	—	—	$(3m+3) \times 2^m$