Square Rooting by Iterative Multiply-Additions

Masayuki ITO †, Naofumi TAKAGI ‡ and Shuzo YAJIMA †

†: Dept. of Information Science, Kyoto University,

Kyoto 606-01, Japan

‡: Dept. of Information Engineering, Nagoya University,

Nagoya 464-01, Japan

Abstract

A new square root algorithm which is performed through iterative multiply-additions is proposed. Although its converging ratio is linear, it is faster than widely used multiplicative methods, such as Newton-Raphson method, for practically available ROM sizes.

Keywords:

Algorithms, computer arithmetic, square rooting, multiply-addition

1 Introduction

A multiply-adder, or a multiply-accumulator, has become standard hardware in digital signal processors. Several microprocessors also have a multiply-adder in them [3]. Hence, it is attractive to develop arithmetic algorithms suitable for a multiply-adder. Square rooting is one of the basic arithmetic operations, so that it is defined in the IEEE Standard for Floating Point Arithmetic [1]. In this contribution, we propose a new algorithm for square rooting which is performed through iterative multiply-additions. We do not assume that we use an existing multiply-adder in a current processor, but assume that we provide a one which is suitable for the proposed algorithm.

Many algorithms have been proposed for square rooting [4]. Newton-Raphson method and Goldschmidt's algorithm are well-known multiplicative algorithms and are widely used in current processors. They start with an initial approximation of the reciprocal of the square root of an operand which is usually read from a ROM table. Then, they refine the approximation by a series of multiplications, subtractions from a constant (complementations) and bit shifts. They have quadratic convergence. Namely, the number of accurate bits of the approximation doubles at each iteration. Each iteration includes three multiplications. Reducing one iteration reduces three multiplications, but causes a huge increase of the required ROM size.

The algorithm to be proposed calculates $\sqrt{X}$ in a completely different way from previous converging algorithms. It does not directly refine the initial approximation of the square root, but refines the value $(\frac{1}{2} - \frac{\sqrt{X}}{K})$ where K is an approximation to $2\sqrt{X}$. It has linear convergence. Namely, the number of accurate bits of the calculated value increases by a constant at each iteration. Each iteration includes only one multiply-addition. After sufficient precision is achieved, one additional multiply-addition yields $\sqrt{X}$. As the number of iterations decreases, the required ROM size increases more gradually than the previous quadratic converging methods. Hence, we can choose an adequate number of iterations according to the available ROM size. For example, IEEE double precision square rooting can be carried out through 7 multiply-additions using a ROM of about 14K bits. Note that the previous quadratic converging methods require three iterations, and hence 9 multiplications, even if this ROM size is available.

2 A New Algorithm

For any constant K which is near to $2\sqrt{X}$, starting with an approximation S_0, the recurrence $S_i := S_{i-1}^2 + (\frac{1}{4} - \frac{X}{K^2})$ leads S_i to $\frac{1}{2} - \frac{\sqrt{X}}{K}$. Note that $\frac{1}{4} - \frac{X}{K^2}$ is invariable. (This recurrence can be derived from the continued fraction expansion of $\sqrt{X}$ [2].)

Let the error of S_{i-1} from $\frac{1}{2} - \frac{\sqrt{X}}{K}$ be ϵ_{i-1}. Namely, let $S_{i-1} = (\frac{1}{2} - \frac{\sqrt{X}}{K}) + \epsilon_{i-1}$. Then, $S_i = (\frac{1}{2} - \frac{\sqrt{X}}{K}) + (1 - \frac{2\sqrt{X}}{K})\epsilon_{i-1} + \epsilon_{i-1}^2$. This implies the converging ratio is linear and its coefficient is $1 - \frac{2\sqrt{X}}{K}$. Therefore, the closer K is to $2\sqrt{X}$, the faster the convergence is.

It is efficient to adopt $\frac{1}{4} - \frac{X}{K^2}$ as S_0, because this value is used in the recurrence. Indeed, this value is a good initial approximation to $\frac{1}{2} - \frac{\sqrt{X}}{K}$ as shown later.

The new algorithm is summarized as follows.

```
[Algorithm SQRT]
Step 1: $S_0 := \frac{1}{4} - L \times X$
Step 2: for $i := 1$ to $k$ do $S_i := S_{i-1} \times S_{i-1} + S_0$
Step 3: $Z := \frac{K}{2} - K \times S_k$                                    □
```

Here, K is an approximation to $2\sqrt{X}$, and is read from a table addressed with several upper bits of X. L is an approximation to $\frac{1}{K^2}$, and must have full precision, i.e., the same precision as Z. It is also read from a table addressed with the same upper bits of X.

We deal with the mantissa part of a floating point number and assume that the operand X is in the range $1 \leq X < 2$. In the case where the exponent part of the original floating point number is odd, we use $K' = \sqrt{2}K$ instead of K in Step 3. K' must have full precision. (We may adopt an additional multiplication by $\sqrt{2}$ instead of keeping K' in a table.)

Each calculation in Steps 1 and 3 and in each iteration of Step 2 is one multiply-addition. $(k+2)$ multiply-additions are required in total.

When the tables are addressed with the m most significant bits of X, in order to minimize the absolute value of the convergence coefficient $1-\frac{2\sqrt{X}}{K}$, the table for K should contain the value $\sqrt{w}+\sqrt{w+2^{-m}}$ for the interval $[w, w+2^{-m})$, where $w=[1.x_1x_2\cdots x_m]$. We round this value at t-th binary place. Then, $|1-\frac{2\sqrt{X}}{K}| < \frac{2^{-m-2}}{w}+\frac{2^{-t-2}}{\sqrt{w}}$ holds. Letting t be m, we get $|1-\frac{2\sqrt{X}}{K}| < 2^{-m-1}$. (Note that m bits are stored for each K when $t=m$, since the integer part of K is always 10 in binary.) This means the number of accurate bits increases by $(m+1)$ bits at each iteration. The initial error ϵ_0 of S_0 from $\frac{1}{2}-\frac{\sqrt{X}}{K}$ satisfies $|\epsilon_0| = \frac{1}{4}(1-\frac{2\sqrt{X}}{K})^2 < 2^{-2m-4}$. Therefore, the error ϵ_k of S_k satisfies $|\epsilon_k| < 2^{-mk-k-2m-4}$. The final error ϵ of Z from $\sqrt{X}$ satisfies $|\epsilon| < 2^{-mk-k-2m-2}$. S_k has to be calculated down to the two more lower binary place than Z.

In summary, we can obtain $\sqrt{X}$ with $mk+k+2m+2$-bit accuracy through k iterations, i.e., through $k+2$ multiply-additions. The sizes of the tables for K, L and K' are $2^m \cdot m$, $2^m \cdot (p-1)$ and $2^m \cdot (p+1)$, respectively, where p is the precision of Z.

Here, we consider square rooting of an IEEE double precision (53-bits) number. Table 1 compares the total required table size for the proposed method with that for Newton-Raphson method, with respect to the number of multiply-additions or multiplications. (In Newton-Raphson method, we assume the final multiplication is combined with the calculations in the final iteration.) When the number of multiply-additions/multiplications is 6, the proposed method requires 28.5K bits, while Newton-Raphson method requires 104K bits. We can reduce the ROM size to 14.1K and 7K bits by means of 7 and 8 multiply-additions, respectively. Note that even if a ROM of 28.5K bits is available, Newton-Raphson method requires 9 multiplications.

Table 1: The required table size for double precision square rooting (bits)

No. of Mul./Mul.-add.	5	6	7	8	9
Proposed	116K	28.5K	14.1K	7K	3.5K
Newton-Raphson	—	104K	—	—	896

It seems easy to provide a multiply-adder which is suitable for the proposed algorithm in current processors by slightly modifying their arithmetic unit.

3 Conclusion

We have proposed a new square rooting algorithm suitable for a multiply-adder, which is faster than widely used multiplicative methods, such as Newton-Raphson method, for several practically available ROM sizes.

Acknowledgement

The authors would like to thank Emeritus Professor I. Ninomiya of Nagoya University for his valuable comments.

References

[1] ANSI/IEEE Std 754-1985, 'Standard for floating point arithmetic'

[2] Ito, M., Takagi, N. and Yajima, S.: 'A high-speed square-rooting algorithm using continued fractions,' Technical Report of IEICE, COMP94-35, July 1994 (in Japanese).

[3] Nadehara, K., Hayashida, M., and Kuroda, I.: 'A low-power, 32-bit RISC processor with signal processing capability and its multiply-adder,' Proc. VLSI Signal Processing VIII, pp. 51–60, Oct. 1995.

[4] Soderquist, P. and Leeser, M.: 'An area/performance comparison of subtractive and multiplicative divide/square root implementations,' Proc. of 12th Symp. on Computer Arithmetic, pp. 132-139, July 1995.