A Pipelined Divider with a Small Lookup Table

CHIN-LONG WEY, SHIN-YO LIN, and MUH-TIAN SHIUE
Department of Electrical Engineering
National Central University
Chung-Li, Taiwan
URL://www.ee.ncu.edu.tw/~clwey

Abstract: - The design of fast dividers is an important issue in high speed computing because division account for a significant fraction of the total arithmetic operation. Taylor series expansion is a well-known multiplicative scheme for high-performance division implementation. This study presents a simple architecture that implements a pipelined divider including the first 6 terms of the Taylor series expansion for approximation. Results show that the developed pipelined divider takes a lookup table of 32B for single precision with a latency of 8.90ns, and 56KB for double precision with 11.46ns, where the circuit is synthesized with TSMC 0.18μm digital CMOS standard cell library.

Key-words: Pipelined Divider, Lookup Table, Taylor Series Expansion, Approximation, Newton-Raphson Algorithm.

1. Introduction

The design of fast dividers is an important issue in high speed computing because division account for a significant fraction of the total arithmetic operation [1]. Most implementations for the division are based on the high-radix SRT algorithm that uses a recurrence producing one quotient digit for each step [2-9]. However, high-performance implementations tend to use multiplier-based convergence schemes, referred to as Multiplicative schemes. Basically, a division (X/Y) is equivalent to a multiplication of X and (1/Y), the reciprocal of the divisor Y. Thus, a multiplier and lookup table(s) for approximating (1/Y) are employed in the multiplicative scheme.

Newton-Raphson and series-expansion algorithms are well-known multiplicative schemes that may require a large lookup table for high-speed operation. For example, the lookup table for a 16-bit seed Newton-Raphson or series-expansion divider is 64KB (bytes) [10]. These 16-bit seed dividers execute a division operation with two iterations in single precision. If 8-bit seed dividers are used, the table size is smaller as 128B, but their latency is longer because they require three iterations. An accurate quotient approximation, which used Taylor-series expansion, has two lookup tables, requiring 400B for single precision [11].

A high-radix pipelinable-division algorithm based on the well-know Taylor series expansion was proposed [12]. The algorithm took the first two terms of the Taylor series expansion for approximation. As illustrated in Figure 1(a), the algorithm provided a simple architecture using a lookup table of 13KB for single precision. It is virtually impossible to implement the divider in double precision because a huge lookup table, nearly 470MB, is required. Therefore, the size of the lookup table is a critical factor to implement a pipelinable divider. Recently, a cost-effective pipelined divider, as shown in Figure 1(b), with a small lookup table was proposed to reduce the table size for implementing in double precision [13]. The algorithm took the first four terms of Taylor series expansion for approximation. Results show that table size can be reduced from 13KB to 208B for single precision, and from 470MB to 56KB for double precision.

Theoretically speaking, including more terms of Taylor series expansion for approximation will improve the accuracy of approximation and thus reducing the size of lookup table. However, including more terms also increases more real-time computations and thus requiring more hardware and longer latency. The question that naturally arises is whether there exists a point where the hardware saving due to the table size reduction is offset the increased hardware due to the need of more real-time computations.

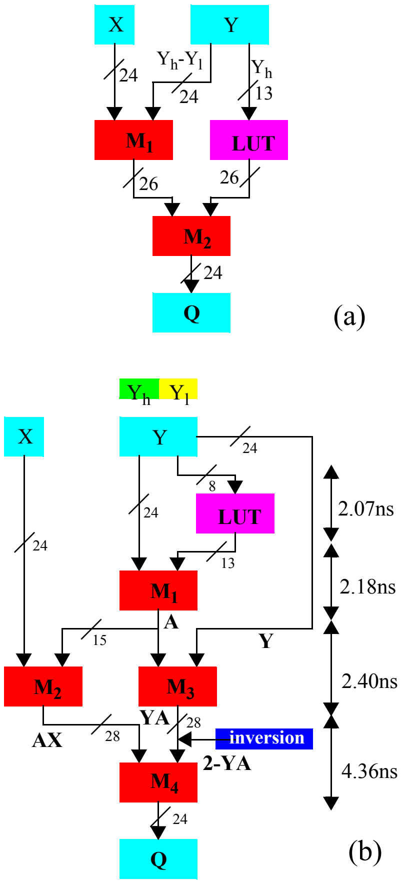

Figure 1. Block Diagrams for Division Algorithms in Single Precision: (a) [12]; and (b) [13].

This study develops a pipelined division algorithm that includes six terms of Taylor series expansion for approximation. Results will show that the table size can be further reduced from 208B in [13] to 36B for single precision, and from 56KB to 1.28K for double precision. The area improvement is achieved at the cost of longer latency, where the latency is increased from 7.18ns to 8.90ns for single precision, from 9.46ns to 11.46ns for double precision. This study also shows that the structure including 6 terms for approximation provides the minimal hardware with a reasonable latency.

In the next section, the basic division algorithm in [13] is reviewed. Section 3 presents the proposed pipelined division algorithms and their hardware implementation. The designs of the division algorithms with four-terms, six terms, and eight terms of Taylor series expansion and their structures are respectively also developed to discuss the design trade-offs. The total error is the errors accumulated at each step of a division algorithm, and it determines the size of lookup table and bit-sizes of the multipliers. The error analysis of each developed algorithm will also presented in this section. Section 4 presents the experimental results. Finally, a concluding remark is given in Section 5.

2. Basic Division Algorithms

Let X and Y be two m-bit normalized fixed-point radicand, where $X=1.x_1x_2..x_m$ and $Y=1.y_1y_2..y_m$. To calculate X/Y, Y is first decomposed into two groups: the higher order bits (Y_h) and the lower order bits (Y_l), i.e., $Y=Y_h+Y_l$, where $Y_h=1.y_1y_2..y_{p-1}$ and $Y_l=0.00..0y_py_{p+1}..y_m$. Therefore, $Y_h >> Y_l$ and Y_l/Y_h is approximately zero. In other words,

$$1 \le X,Y < 2;\ \ 1 \le Y_h < 2\text{-}2^{-p+1};$$
$$\text{and } 0 \le Y_l < 2^{-p+1} \qquad (1)$$

A division operation can be represented by Taylor series expansion as follows:

$$\frac{X}{Y} = \frac{X}{Y_h + Y_l}$$
$$= \frac{X}{Y_h}\left[1 - \frac{Y_l}{Y_h} + \left(\frac{Y_l}{Y_h}\right)^2 - \ldots - \left(\frac{Y_l}{Y_h}\right)^{2n-1} + \ldots\right] \qquad (2)$$

If we take the first two terms of (2) to approximate the division (X/Y), we obtain

$$\frac{X}{Y} \approx \frac{X(Y_h - Y_l)}{Y_h^2} = X(Y_h - Y_l)\left(\frac{1}{Y_h^2}\right) \qquad (3)$$

Therefore, a division operation is executed by multiplying X, (Y_h-Y_l), and ($1/Y_h^2$), where ($1/Y_h^2$) is approximated by a lookup table, and the multiplication of Y(Y_h-Y_l) is computed by a Booth multiplier [12].

The division operation is simplified by first defining a coarse quotient Q'=XA, where A=(Y_h-Y_l)/Y_h^2, and the subdivident X'=X-YQ'. Let Q"=X'(Y_h-Y_l)($1/Y_h^2$)=X'A. The quotient is calculated as

$$X/Y \approx Q' + Q'' = (X + X')A = (2X - YQ')A$$
$$= (AX)(2 - AY) \qquad (4)$$

where both (AY) and (AX) are calculated through parallel multiplications, as shown in Figure 1(b), and (2-AY) is a two's complement of (AY). The approximating error, i.e., the difference between (X/Y) and (Q'+Q"), is expressed as

$$\Delta = \frac{X}{Y} - (2 - AY)AX, \text{ and}$$

$$\Delta = X\left\{\frac{1}{Y} - \left[2 - \frac{Y(Y_h - Y_l)}{Y_h^2}\right]\left[\frac{Y_h - Y_l}{Y_h^2}\right]\right\} = \frac{X}{Y}\left(\frac{Y_l}{Y_h}\right)^4 \quad (5)$$

The size of lookup table and the bit-widths of the multipliers are determined by the possible errors induced. Four possible errors were considered [13]: (a) Error caused by the restriction in the number of entries in the lookup table; (b) Error caused by the bit-width restriction of the lookup table; (c) Error caused by the rounding positions; and (d) Error caused by the bit-inversion.

The first error is calculated by subtracting the actual quotient from the ideal quotient, where the error is expressed as in (5). The number of entries and output bit-width in the lookup table are $2^{(p-1)}$ and q, respectively. Based on the error analysis [13], both p and q are selected from the following inequalities,

$$p > (m+4)/4 \ \text{ and } q \geq -2p+m+5 \quad (6)$$

For single precision with m=24, by (6), p=8 and q=13 are resulted. The table size is 2^{p-1}*q=1664 bits or 208 Bytes, and four multipliers with the dimensions of 24x13, 24x15, 24x15, and 28x28 are employed with a latency of 11.01ns, where the circuit was synthesized with Samsung MDL110 0.25µm CMOS standard cell library [14]. Note that the division algorithm in Figure 1(a) requires a lookup table with the size of 13KB and two multipliers with the dimensions of 26x26 and 24x24, and takes a latency of 7.62ns. On the other hand, for double precision with m=53, by (6), p=15 and q=28 are concluded, where the table size is 448K bits. In addition, the division algorithm requires three 53x28 multipliers and one 58x58 multiplier are employed and takes a latency of 24.2 ns [13].

Unfortunately, the latency of 24.2ns for double precision is impractical. The longer latency is due to the use of table size of 56KB. Thus, it motivates this study to develop a simple, yet better latency for the pipelined division in double precision.

3. Developed Division Algorithms

This section presents the developed pipelined division algorithms.

3.1. Division Algorithms and Hardware Structures

Consider the approximating value Q_{2n} that includes the first 2n terms of the Taylor series expansion in (2), i.e.,

$$\begin{aligned} Q_{2n} &= \frac{X}{Y_h}\left[1 - \frac{Y_l}{Y_h} + \left(\frac{Y_l}{Y_h}\right)^2 - \left(\frac{Y_l}{Y_h}\right)^3 + \ldots - \left(\frac{Y_l}{Y_h}\right)^{2n-1}\right] \\ &= \frac{X}{Y_h}\left[1 - \frac{Y_l}{Y_h}\right]\left[1 + \left(\frac{Y_l}{Y_h}\right)^2 + \left(\frac{Y_l}{Y_h}\right)^4 + \ldots + \left(\frac{Y_l}{Y_h}\right)^{2n-1}\right] \\ &= AX\left[1 + \left(\frac{Y_l}{Y_h}\right)^2 + \left(\frac{Y_l}{Y_h}\right)^4 + \ldots + \left(\frac{Y_l}{Y_h}\right)^{2n-1}\right] \quad (7) \end{aligned}$$

The approximating value can be expressed as in the following theorem,

Theorem 1.

The approximating value Q_{2n} can be expressed as

$$Q_{2n} = Q_{2(n-1)}(1-AY) + Q_2 \quad (8)$$

where Q_2=XA and $A=(Y_h-Y_l)/Y_h^2$.

The approximating value Q_{2n} can also be expressed as

$$\begin{aligned} Q_{2n} &= AX\left[\frac{1-(Y_l/Y_h)^{2n}}{1-(Y_l/Y_h)^2}\right] = AX\left[\frac{1-(Y_l/Y_h)^{2n}}{AY}\right] \\ &= \left(\frac{X}{Y}\right)[1-(Y_l/Y_h)^{2n}] \quad (9) \end{aligned}$$

Therefore, the difference between the ideal quotient and actual quotient, or approximating error, is

$$(X/Y) - Q_{2n} = \left(\frac{X}{Y}\right)\left(\frac{Y_l}{Y_h}\right)^{2n} \quad (10)$$

By (9) & (10), the approximating value and error for n=2 are respectively

$$Q_4=Q_2(1\text{-}AY)+Q_2=Q_2(2\text{-}AY)=AX(2\text{-}AY). \quad (11)$$

$$(X/Y) - Q_4 = (X/Y)(Y_l/Y_h)^4 \quad (12)$$

In other words, the complicated derivations in (4) and (5) can be simply derived from the Taylor series expansion as shown in (11) and (12).

Figure 2 shows an alternative pipelined divider implementing Equation (11). Instead of using a bit-inversion process to find (2-AY) in Figure 1(b), this algorithm calculates the final quotient by adding Q_2=XA to the product of XA and (1-AY), where a simple bit complementer is employed to derive (1-

AY). In practice, the above multiplication and addition may be performed by a MAC (Multiplying and ACcumulating) unit. Both structures in Figures 2 and 1(b) should have the same hardware cost and latency.

Note that the accuracy of the approximating value can be improved by including more terms in (7) for approximation at the cost of more hardware and longer latency for calculating the additional terms. On the other hand, higher accuracy of the approximating value requires smaller size of lookup table. In this study, the table size, total hardware cost, and latency of pipelined division algorithms with n=2, 3, and 4, are presented and evaluated

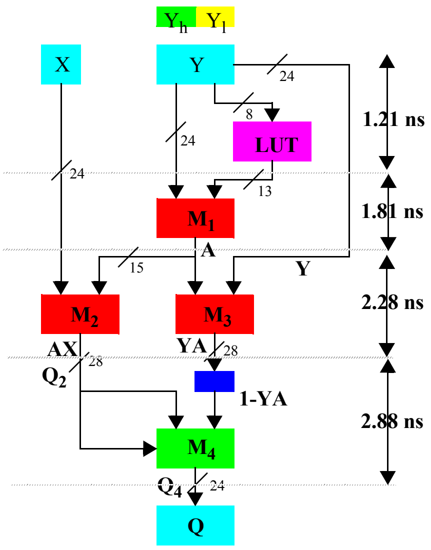

Figure 2. Developed Pipelined Division Algorithm with n=2 in Single Precision:

By theorem 1, the approximating value and error for n=3 are respectively

$$Q_6=Q_4(1-AY)+Q_2 \qquad (13a)$$

$$(X/Y) - Q_6 = (X/Y)(Y_l/Y_h)^6 \qquad (13b)$$

and those for n=4 are

$$Q_8=Q_6(1-YA)+Q_2 \qquad (14a)$$

$$(X/Y) - Q_8 = (X/Y)(Y_l/Y_h)^8 \qquad (14b)$$

For n=3, an additional MAC unit is required, while two MAC units are needed for n=4.

Similar to the error analysis presented in [13], both p and q are determined by the following inequalities,

$$2^{-4p+5} < 2^{-m+1}, \text{ or } \quad p > (m+4)/4 \qquad (15a)$$

$$2^{-2p-q+6} \le 2^{-m+1}, \text{ or } \quad q \ge -2p+m+5 \qquad (15b)$$

$$2^{-2p-m1+5} \le 2^{-3} * 2^{-m+1}, \text{ or } m_1 \ge -2p+m+7 \qquad (15c)$$

$$2^{-m3+2} \le 2^{-3} * 2^{-m+1}, \text{ or } m_2=m_3 \ge m+4 \qquad (15d)$$

Thus, the dimensions of the four multipliers are m x q, m x m_1, m x m_1, and m_2 x m_2, respectively.

For single precision with m=24, by (15a) and (15b), p=8 and q=13 are selected. By (15c), m_1=15, and by (15d), m_2=m_3=28. m_4=24 is the bit-width of the final quotient. Therefore, the dimensions of the multipliers are 24x13, 24x15, 24x15, and 28x28. On the other hand, for double precision with m=53, p=15 and q=28 are resulted. By (15c), m_1=30, and by (15d), m_3=m_2=57. Thus, the dimensions of the multipliers are 53x28, 53x30, 53x30, and 57x57.

3.2. Experimental Results

The pipelined division algorithms in Figures 2have been developed, where the circuits were synthesized with TSMC 0.18mm digital CMOS standard cell library. The pipelined divider with n=2 for single precision requires a lookup table with a size of 2^7x13 or 208B which takes an area of 63,000 μm^2 and a delay of 1.21 ns, as shown in Table 2(a). The dimensions of four multipliers in this divider are 24x13, 24x15, 24x15, and 28x28, respectively, and their corresponding areas are 33,277 μm^2, 41,014 μm^2, 41,014 μm^2, and 86,619 μm^2. This concludes that the total area of the pipelined divider with n=2 for single precision is 264,924μm^2. The critical path of the divider includes the lookup table, M_1, M_3, and M_4, and their corresponding delays are 1.21ns, 1.81ns, 2.28ns, and 2.88ns, respectively. Thus, the divider has a delay of 7.18ns. For double precision, the pipelined divider with n=2, the size of lookup table is 2^{14}x28, or 56KB and takes 207,02,938 μm^2 in area and 1.67ns in delay. The total area of the divider is 21,422,752 μm^2 with a delay of 9.46ns.

4. Conclusions

Taylor series expansion is a well-known multiplicative scheme for high-performance division implementation. A high-radix pipelinable division algorithm based on Taylor series expansion [12] included the first two terms of Taylor series expansion, or n=1, for approximation. It provides a simple architecture with a lookup table of 13KB for single precision. The cost-effective pipelined divider [13] with the first 4 terms of Taylor series expansion, reduced the table size from 13KB to 208B for single precision, and from 470MB to 56KB for double precision. This paper presents a simple architecture that implements a pipelined divider with the first 6 terms of the Taylor series expansion. Results show that the developed pipelined divider further reduces the size of lookup table from 208B to 32B for single precision, and from 56KB to 1.28KB for double precision. This study also shows that the table size can be further reduced as more terms in Taylor series expansion are included for approximation. However, including more terms require additional hardware for calculating the extra terms. As a result, the pipelined divider with the first 6 terms provides an optimal solution which requires the minimum area with a reasonable latency.

Acknowledgment:
This work was supported in part by the Taiwan National Science Council under the grant numbers NSC94-2220-E-008-001, NSC94-2220-E-008-005, NSC94-2220-E-008-008.

References

1. B.K. Bose, L.Pei, G.S. Taylor, and D.A. Patterson, "Fast multiply and divide for VLSI floating-point unit," Proc. IEEE Symp. on Computer Arithmetic, Como, Italy, pp.87-94, May 1987.
2. K. Huang, *Computer Arithmetic: Principles, Architecture, and Design*, Wiley & Sons, Inc., 1979.
3. M.A. Soderstrand, W.K. Jenkins, G.A. Jullien, F.J. Taylor, *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*, IEEE Press, Piscataway, NJ, 1986.
4. I. Koren, *Computer Arithmetic Algorithms*, Prentice-Hall, Inc., New Jersey, 1993.
5. B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, Oxford, 2000.
6. M.D. Ercegovac and T. Lang, *Digital-Recurrence Algorithms and Implementations for Division and Square Root*, Kluwer Academic Publisher, 1993.
7. C.L. Wey and C.P. Wang, "A Fast Radix-4 SRT Divider and Its VLSI Implementation," IEE Proceedings, Computers and Digital Techniques, Vol. 146, No.4, pp.205-210, July 1999.
8 C.L. Wey, "Design of Fast High-Speed SRT Dividers and Their VLSI Implementation," IEE Proceedings, Computers and Digital Techniques, Vol. 147, No.4, pp.275-282, July 2000.
9. P. Komerup, "Digit Selection for SRT Division and Square Root," IEEE Trans. on Computers, Vol. 54, No.3, pp. 294-203, March 2005.
10. S.P. Oberman and M.J. Flynn, "Design Issues in Division and Other Floating Point Operations," IEEE Trans. on Computers, Vol. 46, No.2, pp.154-161, Feb. 1997.
11. D. Wong and M.J. Flynn, "Fast Division Using Accurate Quotient Approximations to Reduce the number of Iterations," IEEE Trans. on Computers, Vol.41, No.8, pp.981-985, August 1992.
12. P. Hung, H. Fahmy, O. Mencer, and M.J. Flynn, "Fast Division Algorithm with a Small Lookup Table," Conference Record, 33rd Asilomar Conference on Signals, Systems, and Controls, Vol. 2, pp. 1465-1468, May 1999.
13. J.-C. Jeong, W.-C. Park, W. Jeong, T.-D. Han, and M.-K. Lee, "A Cost-Effective Pipelined Divider with a Small Lookup Table," IEEE Trans. on Computers, Vol. 53, No. 4, pp.489-495, April 2004.
14. Samsung Electronics Co. Ltd, MDL110 0.25um 2.5V CMOS Standard Cell Library for Pure Logic/MDL Products, 1999.