Fast Division Algorithm with a Small Lookup Table

Patrick Hung, Hossam Fahmy, Oskar Mencer, Michael J. Flynn

Computer Systems Laboratory
Stanford University, CA 94305
email: {hung, hfahmy, oskar, flynn}@arithmetic.stanford.edu

Abstract

This paper presents a new division algorithm, which requires two multiplication operations and a single lookup in a small table. The division algorithm takes two steps. The table lookup and the first multiplication are processed concurrently in the first step, and the second multiplication is executed in the next step. This divider uses a single multiplier and a lookup table with $2^m(2m+1)$ bits to produce $2m$-bit results that are guaranteed correct to one ulp. By using a multiplier and a 12.5KB lookup table, the basic algorithm generates a 24-bit result in two cycles.

1. Introduction

Division is an important operation in many areas of computing, such as signal processing, computer graphics, networking, numerical and scientific applications. In general, division algorithms may be divided into five categories: digit recurrence, functional iteration, high radix, table lookup, and variable latency. These algorithms differ in overall latency and area requirements. An overview of division algorithms can be found in [4].

This paper introduces a new high radix division algorithm based on the well-known Taylor series expansion. A number of high radix division algorithms were also proposed in the past based on the Taylor series. For example, Farmwald [2] proposed using multiple tables to look up the first few terms in the Taylor series. Later, Wong [5] proposed an elaborate iterative quotient approximation with multiple lookup tables. Wong demonstrated that only the first two terms in the Taylor series are necessary to achieve fast division because of the time to evaluate all the power terms.

The previous algorithms consider each individual term in the Taylor series separately; hence, many lookup tables are needed and the designs are complicated. Our proposed algorithm combines the first two terms of Taylor series together, and only requires a small lookup table to generate accurate results. This algorithm achieves fast division by multiplying the dividend in the first step, which is done in parallel with the table lookup. In the second step, another multiplication operation is executed to generate the quotient.

2. Basic Algorithm

Let X and Y be two $2m$-bit fixed point numbers between one and two defined by Equations 1 and 2 where $x_i, y_i \in \{0,1\}$.

$$X = 1 + 2^{-1}x_1 + 2^{-2}x_2 + \ldots + 2^{-(2m-1)}x_{2m-1} \quad (1)$$
$$Y = 1 + 2^{-1}y_1 + 2^{-2}y_2 + \ldots + 2^{-(2m-1)}y_{2m-1} \quad (2)$$

To calculate X/Y, Y is first decomposed into two groups: the higher order bits (Y_h) and the lower order bits (Y_l). Y_h contains the $m+1$ most significant bits and Y_l contains the remaining $m-1$ bits.

$$Y_h = 1 + 2^{-1}y_1 + \ldots + 2^{-(m-1)}y_{m-1} + 2^{-m}y_m \quad (3)$$
$$Y_l = 2^{-(m+1)}y_{m+1} + \ldots + 2^{-(2m-1)}y_{2m-1} \quad (4)$$

The range of Y_h is between 1 and Y_{hmax} ($= 2 - 2^{-m}$), and the range of Y_l is between 0 and Y_{lmax} ($= 2^{-m} - 2^{-(2m-1)}$). Dividing X by Y, we get Equations 5 and 6. Since $Y_h > 2^m \cdot Y_l$, the maximum fractional error in Equation 6 is less than 2^{-2m} (or 1/2 ulp).

$$\frac{X}{Y} = \frac{X}{Y_h + Y_l} = \frac{X(Y_h - Y_l)}{Y_h^2 - Y_l^2} \quad (5)$$
$$\frac{X}{Y} \approx \frac{X(Y_h - Y_l)}{Y_h^2} \quad (6)$$

Using Taylor series, Equation 5 can be expanded at Y_l/Y_h as in Equation 7. The approximation in Equation 6 is equivalent to combining the first two terms in the Taylor series.

$$\frac{X}{Y_h + Y_l} = \frac{X}{Y_h}\left(1 - \frac{Y_l}{Y_h} + \frac{Y_l^2}{Y_h^2} - \ldots\right) \quad (7)$$

Figure 1 shows the block diagram of the algorithm. In the first step, the algorithm retrieves the value of $1/Y_h^2$ from a lookup table and multiplies X with $(Y_h - Y_l)$ at the same time. In the second step, $1/Y_h^2$ and $X \cdot (Y_h - Y_l)$ are multiplied together to generate the result.

2.1. Lookup Table Construction

To minimize the size of the lookup table, the table entries are normalized such that the most significant bit (MSB) of each

0-7803-5700-0/99/$10.00©1999 IEEE

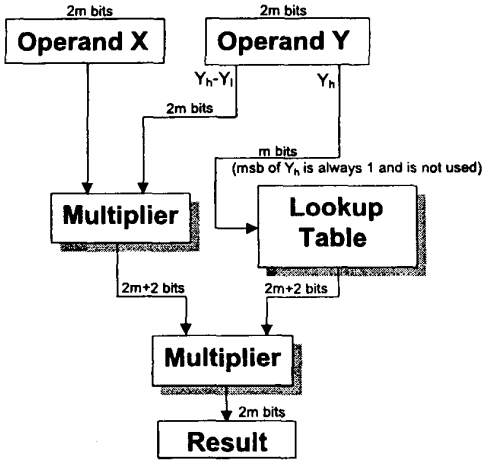

Figure 1: Basic Algorithm

entry is one. These MSB's are therefore not stored in the table.

A lookup table with m=3 is shown in Table 1. $\lfloor 1/Y_h^2 \rfloor$ represents the truncated value of $1/Y_h^2$ to $2m+2$ significant bits. The exponent part of the $1/Y_h^2$ may be stored in the same table, but can also be determined by some simple logic gates. In this example, the exponent is 1.00 when $y_1 = y_2 = y_3 = 0$, the exponent is 0.10 when $y_1 = 0$ and $y_2 \vee y_3 = 1$, the exponent is 0.01 when $y_1 = 1$.

Table 1: A simple lookup table example ($m = 3$)

Y_h	$\lfloor 1/Y_h^2 \rfloor$	Table entry
1.000	1.0000000 × 1.00	0000000
1.001	1.1001010 × 0.10	1001010
1.010	1.0100011 × 0.10	0100011
1.011	1.0000111 × 0.10	0000111
1.100	1.1100011 × 0.01	1100011
1.101	1.1000001 × 0.01	1000001
1.110	1.0100111 × 0.01	0100111
1.111	1.0010001 × 0.01	0010001

2.2. Booth Encoding

Booth encoding algorithm [1] has widely been used to minimize the number of partial product terms in a multiplier. In our division algorithm, special Booth encoders are needed to achieve the $X(Y_h - Y_l)$ multiplication without explicitly calculating the value of $(Y_h - Y_l)$. Lyu and Matula [3] proposed a general redundant binary booth recoding scheme. In our case, the Y_h and Y_l bits are non-overlapping, and a cheaper and faster encoding scheme is feasible.

We use Booth 2 encoding to illustrate our encoding algorithm, but the same principle can apply to the other Booth encoding schemes. In Booth 2 encoding, the multiplier is partitioned into overlapping strings of 3 bits, and each string is used to select a single partial product.

Unlike conventional Booth 2 encoding, the encoding of $(Y_h - Y_l)$ consists of four types of encoders. Figure 2 shows the locations of these four types of encoders: the Y_l group contains all the 3-bit strings that reside entirely within Y_l; the boundary string contains some Y_l bits as well as some Y_h bits; the first Y_h string is located next to the boundary string; the Y_h group contains all the remaining strings within Y_h.

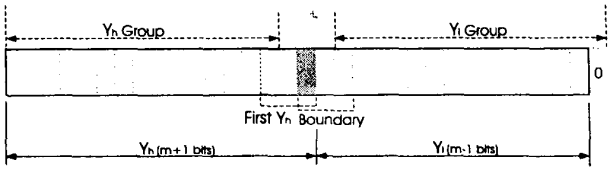

Figure 2: Booth Encoding

The Y_h bits represent positive numbers, whereas the Y_l bits represent negative numbers. Hence, conventional Booth 2 encoding is used in the Y_h group but the partial products in the Y_l group are negated. As shown in the diagram, the boundary region between Y_h and Y_l requires two additional special encoders. Depending on whether m is even or odd, the encoding schemes for these two encoders are different. It is possible that only one such encoder is used in the boundary region, but it implies that this encoder needs to generate $-3\times$ multiplicand (for even m). In order to speed up the multiplication and simplify the encoding logic, two special encoders are used to avoid the "difficult" multiples.

Table 2 summarizes the four different encoding schemes for both even and odd m. It is important to note that the first Y_h encoder actually needs to examine both the first Y_h string and the boundary string when m is odd. If the boundary string is 101, the LSB of the first Y_h string is set to 0 instead of 1. If the boundary string is not 101, the LSB of the first Y_h string is set to be the MSB of the boundary string (as usual). This encoding scheme uses all but two normal Booth encoders and is particularly useful if the same multiplier hardware is used for both the first and the second multiplications.

2.3. Error Analysis

There are four sources of errors: Taylor series approximation error (E_∞), lookup table rounding error (E_T), the rounding error of the first multiplication (E_{M1}), and the rounding error of the second multiplication (E_{M2}).

The total error is equal to $E_\infty + E_T + E_{M1} + E_{M2}$. To minimize this error, the divider can be designed such that $E_\infty \leq 0$, $E_T \leq 0$, $E_{M1} \leq 0$, and $E_{M2} \geq 0$. This means that the table entries are truncated to $2m+2$ bits, the first multiplication is truncated to $2m+2$ bits, and the second multiplication is rounded up to $2m$ bits.

Table 2: Booth Encoding of $(Y_h - Y_l)$

Bits	Y_h Group	Y_l Group	Boundary even m	Boundary odd m	First Y_h even m	First Y_h odd m
00 0	0	0	0	0	0	0
00 1	+1	−1	−1	−1	0	+1
01 0	+1	−1	−1	+1	+1	+1
01 1	+2	−2	−2	0	+1	+2
10 0	−2	+2	+2	−2	−2	−2
10 1	−1	+1	+1	+1	−2	−1
11 0	−1	+1	+1	−1	−1	−1
11 1	0	0	0	−2	−1	0

The Taylor series approximation error (E_∞) is determined by Equation 8. This error is most significant for large Y_l and small Y_h. The maximum approximation error $|E_\infty|$ is slightly less than 1/2 ulp when $Y_l = Y_{lmax}$ and $Y_h = 1$.

$$E_\infty = \frac{X(Y_h - Y_l)}{{Y_h}^2} - \frac{X}{Y} = -\frac{X \cdot {Y_l}^2}{Y \cdot {Y_h}^2} \quad (8)$$

The lookup table has $2m+2$ significant bits, so the maximum truncation error $|E_T| < 1/4$ ulp. Similarly, the maximum rounding error for the first multiplication $|E_{M1}| < 1/4$ ulp, and the maximum rounding error for the second multiplication $|E_{M2}| < 1$ ulp. Thus, the maximum positive error is less than 1 ulp ($|E_{M2}|$), and the maximum negative error is also less than 1 ulp ($|E_\infty + E_T + E_{M1}|$).

3. Optimization Techniques

This section describes two optimization techniques for the division algorithm. The first technique uses a slightly different lookup table and allows the two multiplications to use the same rounding mode, whereas the second technique uses an error compensation term to further reduce the Taylor series approximation error.

3.1. Alternative Lookup Table

As described in Section 2.3, the rounding modes of the first and the second multiplications are different. This may be undesirable if the two multiplications need to share the same multiplier. A simple solution is to use round-to-nearest mode in the two multiplications as well as in constructing the lookup table. Since the error terms can either be positive or negative, the maximum total error becomes the sum of the maximum of each error term.

Let $T(Y_h)$ be the table entry at Y_h with infinite precision. The expression for the approximation error E_∞ is shown in Equation 9 below.

$$E_\infty = X(Y_h - Y_l)T(Y_h) - \frac{X}{Y_h + Y_l} \quad (9)$$

In order to minimize $|E_\infty|$, $T(Y_h)$ is set to be slightly larger than $1/Y_h^2$. For each Y_h, the optimum table entry is determined by setting the maximum positive error (at $Y_l = 0$) to be the same as the maximum negative error (at $Y_l = Y_{lmax}$). Equation 10 shows the expression for the optimum table entry $T(Y_h)_{opt}$.

$$T(Y_h)_{opt} = \frac{2Y_h + Y_{lmax}}{Y_h(Y_h + Y_{lmax})(2Y_h - Y_{lmax})} \quad (10)$$

The approximation error is at its maximum when $Y_h = 1$ and $Y_l = Y_{lmax}$. Using Equation 9, the maximum approximation error can easily be derived as in Equation 11. In this case, $|E_\infty|$ is slightly less than 1/4 ulp.

$$|E_\infty| = \frac{X \cdot Y_{lmax}^2}{(1 + Y_{lmax})(2 - Y_{lmax})} \quad (11)$$

Using round-to-nearest rounding mode, $|E_{M1}| < 1/8$ ulp, $|E_{M2}| < 1/2$ ulp, and $|E_T| < 1/8$ ulp. As in Section 2.3, the total error of the alternative lookup table is also less than 1 ulp. Table 3 illustrates the same example shown in Section 2.1 with the alternative lookup table, where $RN[T(Y_h)]$ represents the round-to-nearest value of $T(Y_h)$ to $2m+2$ significant bits.

Table 3: Alternative Lookup Table ($m = 3$)

Y_h	$RN[T(Y_h)]$	Table entry
1.000	1.0000001 × 1.00	0000001
1.001	1.1001011 × 0.10	1001011
1.010	1.0100101 × 0.10	0100101
1.011	1.0001000 × 0.10	0001000
1.100	1.1100100 × 0.01	1100100
1.101	1.1000010 × 0.01	1000010
1.110	1.0101000 × 0.01	0101000
1.111	1.0010010 × 0.01	0010010

3.2. Error Compensation

The Taylor series approximation error can be further reduced by adding an error compensation term in the first multiplication. Equation 8 shows that the magnitude of the Taylor series approximation error (E_∞) increases when either Y_l gets larger or Y_h gets smaller. By looking at the first few bits of Y_l and Y_h, it is possible to identify large Y_l and small Y_h, and then compensate for the approximation error. However, it is important to ensure that the approximation error is not overcompensated and becomes positive; otherwise this would increase the total error (Section 2.3).

Figure 3 depicts a simple error compensation scheme. In this diagram, $C(Y)$ represents the positive error compensation that is used to correct the negative Taylor series approximation error. The new approximation equation becomes:

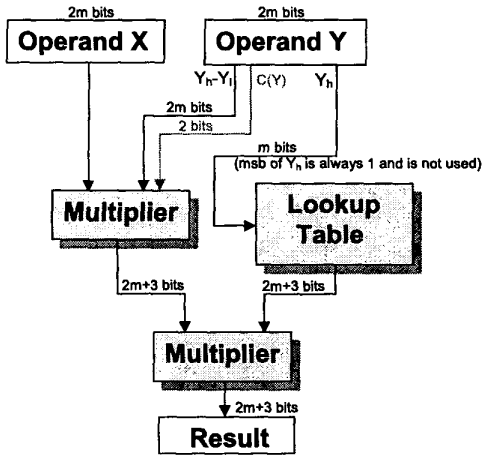

Figure 3: Error Compensation

$$\frac{X}{Y} \approx \frac{X(Y_h - Y_l + C(Y))}{Y_h^2} \quad (12)$$

The expression for $C(Y)$ is shown in Equation 13. Depending on the first few terms of Y_h and Y_l, $C(Y)$ is set to 0, $2^{-(2m+2)}$, or $2^{-(2m+1)}$.

$$C(Y) = 2^{-(2m+2)} \cdot y_{m+1} \cdot y_{m+2}(1 + \overline{y_1} \cdot \overline{y_2} \cdot y_{m+3}) \quad (13)$$

The error compensation only requires an additional partial product in the multiplier and a few simple logic gates (shown in Equation 13). The maximum Taylor series approximation error is 9/32 ulp when $Y_h = 1$ and $Y_l = (3Y_{lmax} - 1)/4$. Figure 4 shows the approximation error with different Y_l.

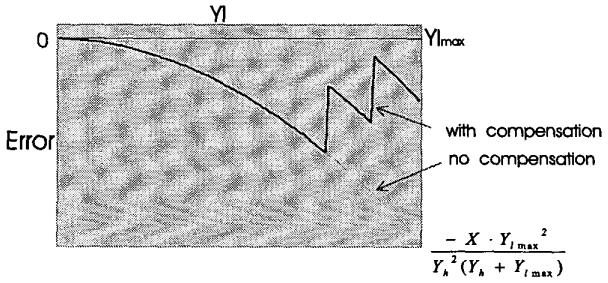

Figure 4: Approximation Error with Error Compensation

A similar error compensation scheme can also be used for the alternative lookup table shown in Section 3.1. The expressions for the new approximation and the error compensation are determined by Equations 14 and 15, respectively. In this case, $C(Y)$ is set to be $-2^{-(2m+2)}$, 0, or $2^{-(2m+2)}$ depending on the first few bits in Y_h and Y_l.

$$\frac{X}{Y} \approx \frac{(2Y_h + Y_{lmax})(Y_h - Y_l + C(Y))X}{Y_h(Y_h + Y_{lmax})(2Y_h - Y_{lmax})} \quad (14)$$

$$C(Y) = 2^{-2m-2}(\overline{y_1} \cdot y_{m+1} \cdot y_{m+2} - \overline{y_{m+1}}) \quad (15)$$

4. Discussion

This paper presents a simple and fast division algorithm based on Taylor series expansion. Using a multiplier and a lookup table with $2^m(2m+1)$ bits, this algorithm produces a $2m$-bit result in two steps. For example, a 12.5KB lookup table is required for single precision (24 bits) floating point division.

The same principle can be applied to some elementary functions, such as square root. Using the same definitions of Y, Y_h, and Y_l as before, we get the following approximation.

$$\sqrt{Y} = \frac{Y}{Y_h^{\frac{1}{2}}}(1 - \frac{Y_l}{2Y_h} + \frac{3Y_l^2}{8Y_h^2} - \ldots) \approx \frac{Y(Y_h - Y_l/2)}{Y_h^{\frac{3}{2}}} \quad (16)$$

This is very similar to the approximation used in the division algorithm. The differences are that the Y_l term is shifted by one bit in the numerator and the lookup table contains $1/Y_h^{\frac{3}{2}}$ entries instead of $1/Y_h^2$ entries.

We can also combine more than the first two terms in the Taylor series expansion. For example, if we use the first four terms in the expansion, we get the following approximation.

$$\frac{X}{Y} \approx \frac{X(Y_h - Y_l)}{Y_h^2}(1 + \frac{Y_l^2}{Y_h^2}) \quad (17)$$

As before, only a single $1/Y_h^2$ lookup table is needed but this algorithm also needs to calculate Y_l^2. Our next step is to generalize the existing algorithm and to investigate the optimum Taylor series approximation for different input precisions.

5. References

[1] A. D. Booth. A signed binary multiplication technique. *Quarterly Journal of Mechanics and Applied Mathematics*, pages 236–240, June 1951.

[2] P. M. Farmwald. *On the design of high performance digital arithmetic units.* PhD thesis, Stanford University, 1981.

[3] C.N. Lyu and D. Matula. Redundant Binary Booth Recoding. In *Proceedings of IEEE Symposium on Computer Arithmetic*, pages 50–57, July 1995.

[4] Stuart F. Obermann and Michael J. Flynn. Division algorithms and implementations. *IEEE Transactions on Computers*, 46(8):833–854, August 1997.

[5] Derek Wong and Michael J. Flynn. Fast division using accurate quotient approximations to reduce the number of iterations. *IEEE Transactions on Computers*, pages 981–995, August 1992.