Efficient Multiprecision Floating Point Multiplication with Optimal Directional Rounding

Werner Krandick*

Research Institute for Symbolic Computation
Johannes Kepler University
A-4040 Linz, Austria
krandick@risc.uni-linz.ac.at

Jeremy R. Johnson†

Mathematics and Computer Science
Drexel University
Philadelphia, PA 19104
jjohnson@king.mcs.drexel.edu

Abstract

An algorithm is described for multiplying multiprecision floating point numbers. The algorithm produces either the smallest floating point number greater than or equal to the true product or the greatest floating point number smaller than or equal to the true product. Software implementations of multiprecision precision floating point multiplication can reduce the computing time by a factor of two if they do not compute the low order digits of the product of the two mantissas. However, these algorithms do not necessarily provide optimally rounded results. The algorithm described in this paper is guaranteed to produce optimally rounded results and typically obtains the same savings.

1 Introduction

We present an algorithm for multiplying multiprecision floating point numbers. The algorithm is intended to support an arbitrary precision interval arithmetic package [5]. As such it has to produce results rounded in the appropriate direction. Our algorithm returns either the smallest floating point number greater than or equal to the true product or the greatest floating point number smaller than or equal to the true product. A rounding operation which satisfies this requirement is called an *optimal directional rounding.* Optimal directional rounding provides a well defined, implementation independent, semantics for floating point arithmetic. For this reason, floating point arithmetic with optimal directional rounding has been advocated even if it is more costly than algorithms which do not produce optimally rounded results [6]. The IEEE standard 754 for hardware binary floating point arithmetic supports optimal directional rounding [4]. Optimal directional rounding can easily be accomplished for floating point multiplication if the full product of the mantissas is computed. Floating point multiplication with directional rounding can be made more efficient if optimal rounding is not used. Our algorithm produces optimal roundings yet typically requires only half as much computing time as the full product.

We use the "short product", outlined in Exercise 15 in Section 4.3.1 of Knuth [6], which uses classical multiplication but suppresses the computation of low order digits when the two mantissas are multiplied. This method will asymptotically save about half of the computing time if both operands and the result have the same precision. The short product can also be used in conjunction with Karatsuba's algorithm for fast integer multiplication. Other multiprecision packages either use the full product (Maple V [2], Reduce [8]), or they use the short product (MPFUN [1]) but do not provide the accuracy specified above.

The short product computes $M+2$ digits, where M is the precision of the result. Our algorithm is based on an efficient test to determine if the result obtained by rounding the short product is the same as that obtained by rounding the full product. If the test succeeds then we can return the optimally rounded result using the short product instead of the full product. If the test fails we must resort to the full product to compute the optimal directional rounding. We show that if the radix used to represent the mantissas is sufficiently large, the test will almost always succeed. The radix will be sufficiently large in a software implementation of multiprecision floating point arithmetic where the radix is typically chosen to be as close to

*Supported by the Austrian Science Foundation (Grant M022-PHY).

†Supported by NSF-grant CCR-9211016.

the largest single precision integer as possible.

The test first determines if the significant digits of the short product are the same as those that would have been supplied by the full product. This test is performed using the trailing digits supplied by the short product and can be done in constant time. In almost all cases the test will be able to indicate an affirmative answer. If this part of the test succeeds, we need to determine if the true product is exactly representable as a floating point number with the desired precision. This will be the case if the trailing two digits of the short product were zero and all of the suppressed digit products were also zero. Thus we need to determine if the suppressed digits were all zero without actually computing the full product. This can be done by counting the number of trailing zeros in the input operands. This part of the test is similar to an algorithm proposed by Santoro, Berwick, and Horowitz in [7] for computing the sticky bit.

Section 2 reviews basic notation that is used in the paper. Section 3 gives a concise presentation of the main idea. Section 4 supplies rigorous definitions and proofs for the claims made in Section 3. Finally, Section 5 presents an empirical comparison of floating point multiplication with and without the use of the short product.

2 Notation

Let a be a non-zero integer; let β be an integer > 1. The representation of a with respect to base β has *β-length*

$$L_\beta(a) = \lfloor \log_\beta |a| \rfloor + 1.$$

In the sequel β will be assumed to be a power of 2, namely

$$\beta = 2^\zeta \text{ with } \zeta \geq 3. \tag{1}$$

Typically ζ depends on the wordsize of the computer. Our algorithm requires $\zeta \geq 3$. For all non-zero integers a

$$L_\beta(a) = \lceil L_2(a)/\zeta \rceil.$$

Definition 1 *An integer $a \neq 0$ is called* normalized *if*

$$L_\beta(a) = L_2(a)/\zeta.$$

Lemma 1 *The following conditions are equivalent for non-zero integers a:*

1. *a is normalized.*
2. *The high-order bit in the high-order β-digit of a is 1.*
3. *With $n = L_\beta(a)$,*
$$\frac{\beta}{2}\beta^{n-1} \leq |a| < \beta^n.$$
4. *With $n = L_\beta(a)$,*
$$2^{\zeta n - 1} \leq |a| < 2^{\zeta n}.$$

Definition 2 *The β-*order *$o_\beta(a)$ of an integer a is defined as*

$$o_\beta(a) = \max\{k \mid \beta^k | a\}$$

The β-order of an integer is the number of trailing zeros in its representation with respect to base β.

For convenience all integers occurring in this paper will assumed to be positive. Of course, the results carry over to arbitrarily signed integers. Furthermore, A and B will always stand for positive integers consisting of n_1 and n_2 β-digits, respectively:

$$A = \sum_{h=1}^{n_1} a_h \beta^{h-1} \quad \text{and} \quad B = \sum_{j=1}^{n_2} b_j \beta^{j-1}, \quad A, B > 0.$$

3 The main idea

This section outlines the main idea. For precise definitions and proofs see Section 4. Let two floating point numbers be given with normalized mantissas A and B and with precisions $L_\beta(A) = n_1$ and $L_\beta(B) = n_2$. We want to compute the floating point product to precision $M = \max(n_1, n_2)$. Let $N = \min(n_1, n_2)$. The "short product"

$$C = \sum_{h+j \geq N} a_h b_j \beta^{h+j-N}.$$

can be computed using just

$$n_1 n_2 - (\frac{N^2 - 3N}{2} + 1)$$

digit multiplications (Lemma 4). C will have $M + 2$ β-digits, where $M = \max(n_1, n_2)$ (Proposition 1).

$$C = \sum_{k=1}^{M+2} c_k \beta^{k-1},$$

where the c_k's are β-digits and $c_{M+2} \neq 0$.

The short product can be visualized by looking at the usual pencil and paper method for multiplying integers. Figure 1 illustrates the case $n_1 \leq n_2$ with $N = \min(n_1, n_2)$ and $M = \max(n_1, n_2)$. The indices of the digit products in each column sum to j, for

$$
\begin{array}{ccccccc|ccccc}
 & & & & b_M & \cdots & b_N & b_{N-1} & & \cdots & & b_1 \\
 & & & & & & a_N & a_{N-1} & & \cdots & & a_1 \\
\hline
 & & & \cdots & b_{N+1}a_1 & b_N a_1 & b_{N-1}a_1 & b_{N-2}a_1 & \cdots & b_3a_1 & b_2a_1 & b_1a_1 \\
 & & & \cdots & b_N a_2 & b_{N-1}a_2 & b_{N-2}a_2 & b_{N-3}a_2 & \cdots & b_2a_2 & b_1a_2 & \\
 & & & \cdots & b_{N-1}a_3 & b_{N-2}a_3 & b_{N-3}a_3 & b_{N-4}a_3 & \cdots & b_1a_3 & & \\
 & & & \cdots & \vdots & \vdots & \vdots & \vdots & \cdots & & & \\
 & & & \cdots & b_4a_{N-2} & b_3a_{N-2} & b_2a_{N-2} & b_1a_{N-2} & & & & \\
 & & b_M a_{N-1} & \cdots & b_3a_{N-1} & b_2a_{N-1} & b_1a_{N-1} & & & & & \\
 & b_M a_N & b_{M-1}a_N & \cdots & b_2a_N & b_1a_N & & & & & & \\
\hline
c_{M+2} & c_{M+1} & c_M & \cdots & c_3 & c_2 & c_1 & & & & &
\end{array}
$$

Resulting Mantissa

Figure 1: Illustration of the short product for the case $n_1 \le n_2$ with $N = \min(n_1, n_2)$ and $M = \max(n_1, n_2)$.

$j = 2, \ldots, M+N$. The error obtained by suppressing the low order digit products (those to the right of the column with $j = N$) is $\le (N-2)\beta^{N-1}$ (Lemma 2). Therefore the M leading β-digits of the product are the same as the M leading β-digits of the short product if the error term added to c_2 does not produce a carry, corrupting c_3. Moreover, it will be shown (Proposition 2) that the leading $M\zeta$ binary digits of the product are the same as the leading $M\zeta$ binary digits of the short product if the error term added to c_2 does not produce a carry into the two high order bits of c_2. More explicitly, let γ be the number formed by the $\zeta - 2$ low order bits of c_2. Clearly, $\gamma < \frac{\beta}{4}$. If

$$\gamma + (N-2) < \frac{\beta}{4}$$

then the $M\zeta$ leading binary digits of C agree with the $M\zeta$ leading binary digits of AB, i.e. the M β-digits obtained by normalizing C are the same as those that would have been obtained by normalizing AB.

The above condition will not be satisfied if

$$\gamma \in \left\{ \frac{\beta}{4} - 1, \ldots, \frac{\beta}{4} - (N-2) \right\}$$

Assuming that all $\frac{\beta}{4}$ bit-combinations of length $\zeta - 2$ occur with equal likelihood as values of γ, the probability that the condition is not satisfied is

$$p = \frac{4(N-2)}{\beta}.$$

In practice this number will be very small. For example, if $N = 50$ and $\beta = 2^{29}$ then $p < 10^{-6}$.

Assume now that the $M\zeta$ leading bits of C have been used to construct a normalized floating point number c of precision M. Let γ' be the number formed by the remaining bits of C. For an explanation of the rounding procedure assume that $c > 0$ and that the number to be returned is "the smallest floating point number c' of precision M which is greater than or equal to the true result". Clearly, the number c' will either be c itself or c^+, the neighboring floating point number to the right of c. If $\gamma' \ne 0$, $c' = c^+$. If $\gamma' = 0$, c' will be c if and only if the suppressed digit products were all zero. The latter is the case if and only if

$$o_\beta(A) + o_\beta(B) \ge N - 2$$

(Proposition 3). So the result returned is

$$c' = \begin{cases} c^+ & \text{if } \gamma' \ne 0 \text{ or } o_\beta(A) + o_\beta(B) < N-2 \\ c & \text{otherwise.} \end{cases}$$

4 Definitions and proofs

Definition 3 *The* short product *of A and B with respect to an integer N is defined as*

$$
\begin{aligned}
C &= A \times_N B \\
&= \begin{cases} \sum_{h+j \ge N} a_h b_j \beta^{h+j-N} & \text{if } 2 \le N \le n_1 + n_2 \\ A \times_2 B & \text{if } N < 2 \\ 0 & \text{if } N > n_1 + n_2 \end{cases}
\end{aligned}
$$

Remark 1 *The short product is a generalization of the ordinary product. It is obtained by suppressing certain digit products when multiplying A and B. Here are some useful elementary properties of the short product.*

1. *For $N \le 2$, $A \times_N B = AB$.*
2. *$A \times_{(n_1+n_2)} B = a_{n_1} b_{n_2}$.*
3. *For $2 \le M \le N \le n_1 + n_2$, $\beta^{N-2}(A \times_N B) \le \beta^{M-2}(A \times_M B)$.*

4. *If $A \le \tilde{A}$ then $A \times_N B \le \tilde{A} \times_N B$ for all N.*
5. *For all N, $A \times_N B = B \times_N A$.*

Proposition 1 *Let $2 \le N \le n_1 + n_2$. The length of $A \times_N B$ is*

1. *in general*

$$n_1 + n_2 - N + 1 \le L_\beta(A \times_N B) \le n_1 + n_2 - N + 2,$$

2. *for normalized A and B*

$$L_\beta(A \times_N B) = n_1 + n_2 - N + 2,$$

and

$$\zeta(n_1 + n_2 - N + 1) + (\zeta - 2) \le L_2(A \times_N B) \le \zeta(n_1 + n_2 - N + 2).$$

Proof. Using the statements of Remark 1 the first inequality in statement 1 is obtained as follows:

$$\begin{aligned} \beta^{N-2}(A \times_N B) &\ge \beta^{n_1+n_2-2}(A \times_{(n_1+n_2)} B) \\ &\ge \beta^{n_1+n_2-2}(\beta^{n_1-1} \times_{(n_1+n_2)} \beta^{n_2-1}) \\ &= \beta^{n_1+n_2-2}, \end{aligned} \tag{2}$$

and so, $L_\beta(A \times_N B) \ge (n_1 + n_2 - 1) - (N - 2) = n_1 + n_2 - N + 1$.

The second inequality in statement 1 is proven similarly:

$$\begin{aligned} \beta^{N-2}(A \times_N B) &\le \beta^0(A \times_2 B) \\ &= AB \\ &\le (\beta^{n_1} - 1)(\beta^{n_2} - 1) \\ &< \beta^{n_1+n_2}, \end{aligned}$$

and so, $L_\beta(A \times_N B) \le (n_1 + n_2) - (N - 2) = n_1 + n_2 - N + 2$.

If A and B are normalized, Inequality (2) can be strengthened to

$$\beta^{N-2}(A \times_N B) \ge$$

$$\beta^{n_1+n_2-2}(\frac{\beta}{2}\beta^{n_1-1} \times_{(n_1+n_2)} \frac{\beta}{2}\beta^{n_2-1}) = \frac{\beta}{4}\beta^{n_1+n_2-1}.$$

This proves the first inequality in statement 2. □

Definition 4 *The* error ϵ_N *of the short product $A \times_N B$ is defined as*

$$\epsilon_N = \begin{cases} AB - \beta^{N-2}(A \times_N B) & \text{if } 2 \le N \le n_1 + n_2 \\ 0 & \text{if } N < 2 \\ AB & \text{if } N > n_1 + n_2 \end{cases}$$

Remark 2 *The error ϵ_N is the accumulated contribution to AB of the digit products suppressed in $A \times_N B$. For $2 \le N \le n_1 + n_2$*

$$\epsilon_N = \sum_{h+j<N} a_h b_j \beta^{h+j-2}. \tag{3}$$

Lemma 2 *An error bound in case $2 \le N \le n_1 + n_2$ is given by*

$$\epsilon_N \le (N - 2)\beta^{N-1}.$$

Proof. The bound is derived from Equation (3):

$$\begin{aligned} \epsilon_N &= \sum_{h+j<N} a_h b_j \beta^{h+j-2} \\ &\le \sum_{l=2}^{N-1} \sum_{h+j=l} (\beta - 1)^2 \beta^{l-2} \\ &= \sum_{l=2}^{N-1} (l - 1)(\beta - 1)^2 \beta^{l-2} \\ &= \sum_{l=0}^{N-3} (l + 1)(\beta - 1)^2 \beta^l \\ &\le (N - 2)(\beta - 1)^2 \frac{\beta^{N-2} - 1}{\beta - 1} \\ &= (N - 2)(\beta - 1)(\beta^{N-2} - 1) \\ &\le (N - 2)\beta^{N-1} \square \end{aligned}$$

Lemma 3 *Let $2 \le N \le n_1 + n_2$. Then*

$$\beta^{N-2}(A \times_N B) \le AB \le \beta^{N-2}(A \times_N B) + (N-2)\beta^{N-1}.$$

Proof. The first inequality follows from combining statement 1 of Remark 1 with statement 3 (letting $M = 2$). The second inequality follows from Lemma 2 and Definition 4. □

Proposition 2 *Let A and B be normalized, and let $n_1, n_2 \ge 2$. Let $N = \min(n_1, n_2)$, let $M = \max(n_1, n_2)$, and let*

$$C = A \times_N B = \sum_{k=1}^{M+2} c_k \beta^{k-1} \tag{4}$$

with β-digits c_k. Let γ be the number formed by the $(\zeta - 2)$ low order bits of c_2. If

$$\gamma + (N - 2) < \frac{\beta}{4} \tag{5}$$

then the $M\zeta$ leading binary digits of C agree with the $M\zeta$ leading binary digits of AB.

Proof. According to Proposition 1 $L_\beta(C) = M + 2$. Furthermore, Proposition 1 implies

$$L_2(C) \geq \zeta(M+1) + (\zeta - 2),$$

hence at most two high order bits of c_{M+2} are zero. Thus, the $M\zeta$ leading bits of C are in the β-digits $c_{M+2}, \ldots, c_3$ and possibly in the two high order bits of c_2. Inequality 5 thus implies that the $M\zeta$ leading bits of C and $C+(N-2)\beta$ are the same. Therefore the $M\zeta$ leading bits of $\beta^{N-2}C$ and $\beta^{N-2}C+(N-2)\beta^{N-1}$ are the same, and Lemma 3 implies the assertion. □

Proposition 3 *Let* $2 \leq N \leq n_1 + n_2$. *Then*

$$\epsilon_N = 0 \quad \text{if and only if} \quad o_\beta(A) + o_\beta(B) \geq N - 2.$$

Proof. Due to Equation (3) the error ϵ_N vanishes if and only if

$$(a_k b_l \neq 0 \Rightarrow k + l \geq N). \tag{6}$$

It thus suffices to show that (6) is equivalent to $o_\beta(A) + o_\beta(B) \geq N - 2$.

1. Assume (6). Let $\kappa = o_\beta(A)$, $\lambda = o_\beta(B)$. Then $a_{\kappa+1} \neq 0$ and $b_{\lambda+1} \neq 0$, hence $a_{\kappa+1}b_{\lambda+1} \neq 0$, and so, because of (6), $\kappa + \lambda + 2 \geq N$.

2. Assume $o_\beta(A)+o_\beta(B) \geq N-2$. Assume further that $a_k b_l \neq 0$. Then $k \geq o_\beta(A) + 1$ and $l \geq o_\beta(B)+1$, and hence $k+l \geq o_\beta(A)+o_\beta(B)+2 \geq N$. □

Lemma 4 *Computing the short product* $A \times_N B$ *using Equation* (2) *requires*

$$\mu_N = \begin{cases} n_1 n_2 - (\frac{N^2-3N}{2} + 1) & \text{if } 2 \leq N \leq n_1 + n_2 \\ n_1 n_2 & \text{if } N < 2 \\ 0 & \text{if } N > n_1 + n_2 \end{cases} \tag{7}$$

digit multiplications.

Proof. It suffices to show that for $2 < N \leq n_1 + n_2$ the sum on the right hand side of Equation (3) consists of $\frac{N^2-3N}{2} + 1$ terms. For each $l \in \{2, \ldots, N-1\}$ there are $l - 1$ pairs (h, j) such that $h + j = l$, hence the sum in (3) consists of

$$\sum_{l=2}^{N-1} (l-1) = \frac{N-2}{2}(N-1) = \frac{N^2 - 3N + 2}{2}$$

terms. □

The following lemma is useful for implementations of the short product calculation.

Lemma 5 *Assume*

$$2 \leq N \leq n_1 + n_2 \ \wedge \ 1 \leq n_1 \ \wedge \ 1 \leq n_2.$$

Then the lowest index k *such that* a_k *contributes to* $A \times_N B$ *is*

$$k = \max(N - n_2, 1),$$

and the lowest index l *such that* b_l *contributes to* $A \times_N B$ *is*

$$l = \max(N - n_1, 1).$$

Proof. Due to statement 5 of Remark 1 only the first statement needs to be proved. Clearly,

$$k = \min\{\ \kappa \mid 1 \leq \kappa \leq n_1 \wedge \exists\lambda(1 \leq \lambda \leq n_2 \wedge \kappa + \lambda \geq N)\}.$$

Since $\exists\lambda(1 \leq \lambda \leq n_2 \wedge \kappa + \lambda \geq N)$ is equivalent to $1 \leq N - \kappa \leq n_2$,

$$k = \min\{\ \kappa \mid 1 \leq \kappa \leq n_1 \wedge 1 \leq N - \kappa \leq n_2\}.$$

Now let

$$K = \max(N - n_2, 1).$$

It will be shown that $K = k$:

1. If $N - n_2 \leq 1$ then $K = 1$, so $1 \leq K \leq n_1$, $N - K = N - 1$ and, of course, $1 \leq N - 1 \leq n_2$. Obviously K is minimal with $1 \leq K$, so $K = k$.

2. If $N - n_2 > 1$ then $K = N - n_2$, so $1 < K \leq n_1$ and $N - K = n_2$. But clearly $1 \leq n_2 \leq n_2$. Obviously K is minimal with $N - K \leq n_2$, so $K = k$. □

5 An experiment

The performance of two algorithms for floating point multiplication was compared for various precisions N of the operands. Both algorithms used the classical multiplication technique. Algorithm 1 used the full product, algorithm 2 used the short product with respect to N in the way described. Both algorithms were implemented in Portable C using the `saclib`-library [3] of computer algebra programs, and the radix used was $\beta = 2^{29}$. Table 1 lists the computing time ratios

$$\rho = \frac{t_2}{t_1}$$

in percent. In this experiment mantissas were represented as `saclib`-integers, i.e. as lists. In an array implementation one would expect to obtain timing ratios which are closer to the ratios

$$\sigma = \frac{\mu_N}{N^2}$$

Table 1: Speed-up Due to Short Product.

N	ρ	σ
1	139%	100%
2	113%	100%
3	104%	89%
4	98%	81%
5	92%	76%
10	74%	64%
20	63%	57%
30	59%	55%
40	57%	54%
50	56%	53%

of the respective numbers of digit products required by the algorithms. According to Lemma 4 the number μ_N of digit products required by Algorithm 2 was obtained from Equation (7) by letting $n_1 = n_2 = N$.

The table suggests that in our implementation the full product should be used for precisions $\leq$ 4; for higher precisions it is more efficient to use the short product. It is possible to use Karatsuba's method in order to compute the short product. For details of the implementation see [5].

References

[1] David H. Bailey. A portable high preformance multiprecision package. Technical report, NASA Ames Research Center, 1992. RNR Technical Report RNR-90-022.

[2] B. W. Char, K. O. Geddes, G. H. Gonnet, B. L. Leong, M. B. Monagan, and S. M. Watt. *Maple V Language Reference Manual.* Springer-Verlag, New York, 1991.

[3] George E. Collins et al. `saclib` user's guide. Technical Report RISC-Linz Report Series Number 93-19, Research Institute for Symbolic Computation, RISC-Linz, Johannes Kepler University, A-4040 Linz, Austria, 1993.

[4] IEEE. *IEEE Standard 754-1985 for binary floating-point arithmetic*, 1987. Reprinted in SIGPLAN 22,2,9-25.

[5] J. R. Johnson and W. Krandick. A multiprecision floating point and interval arithmetic package for symbolic computation. Technical Report RISC-Linz Report Series Number 93-20, Research Institute for Symbolic Computation, RISC-Linz, Johannes Kepler University, A-4040 Linz, Austria, 1993.

[6] D. E. Knuth. *The Art of Computer Programming*, volume 2. Addison-Wesley Publishing Company, Reading, Massachusetts, 2nd edition, 1981. Seminumerical Algorithms.

[7] Mark R. Santoro, Gary Berwick, and Mark A. Horowitz. Rounding algorithms for IEEE multipliers. In Milos D. Ercegovac and Earl Swartzlander, editors, *Proceedings 9th Symposium on Computer Arithmetic*, pages 176–183. IEEE Computer Society Press, 1989.

[8] T. Sasaki. An arbitrary precision real arithmetic package in REDUCE. In E. W. Ng, editor, *EUROSAM 79, Colloque International sur les Méthodes de Calcul Symbolique et Algébrique, Marseille, France*, number 72 in Lecture Notes in Computer Science, pages 358–368, Berlin-Heidelberg-New York, 1979. Springer-Verlag.