HIGH-SPEED MULTIPLICATION AND MULTIPLE SUMMAND ADDITION

Raymond S. Lim
Ames Research Center, NASA
Moffett Field, California

Summary

The problem of high-speed multiplication is considered from the viewpoint of summand generation and summand summation. The goal is to obtain at least a 2's-complement, 32-bit floating-point (sign plus 24-bit fraction) multiplication in 10 to 20 ns using ECL LSI packages. Summand generation is implemented by mxm-bit multipliers. The optimum values for m are 9, 13, 17, or 21. Summand summation is implemented by a row of (p,2) column-summing counters. The (3,2), (5,2), and (7,2) counters are optimum choices. These counters compress p inputs into two outputs plus nonpropagating carry bits, where these bits are added to the next higher-order stage with at most two full adder delays.

Introduction

In the design of high-performance computers, methods for high-speed multiplication are a continuous research effort. In recent literature, the theory and implementation of various methods have received considerable attention.[1-4] There remains, however, a gap between theory and practice. This gap exists primarily because for high-speed multiplication, the total number of IC chips used (and hence wiring) must be reduced. The effect of wiring on computer arithmetic speed is illustrated by the observation that in present high-performance computers, logic signals spend more than half the time propagating through wires connecting the IC chips. Thus, using faster circuits could at most double the arithmetic speed.

There are two basic problems in high-speed multiplication: summand generation and summand summation. The summand summation problem of adding p n-bit numbers using adders with p inputs and q outputs, called (p,q) counters, was considered by Singh and Waxman,[1] and Ho and Chen.[2] Reference 1 describes a technique of partitioning these p numbers columnwise, such that each column partition contains r bits of each of the p numbers, where r is an integer $\geq \log_2(p-1)$, in which the final sum can be obtained in r + 1 addition cycles. In Ref. 2, (p,q) counters, such as the (7,3) counter, were considered for compressing p summands into q summands by adding each column independently. In Ref. 3, Agrawal and Reddy extended the Dadda scheme to make use of commercially available 4-bit full adders. The addition of p rows requires $\log_2 p$ adding stages. In Ref. 4, Stenzel, et al. considered the multiplication problem of (1) using mxm-bit multipliers for summand generation, and (2) using (p,q) counters for summand summation. They have constructed a 24×24-bit multiplier by using 4×4-bit multiplers, a (5,5,4) counter, and (2,2,2,3,5) counters. Using standard STTL logic, this multipler contains 90 ICs and operates at about 200 ns. In the above four schemes, several adding stages are required to reduce q to 2, which in conclusion is not conducive to high-speed multiplication.

In this paper, the problem of designing an n-bit floating-point (sign plus n-1 bits fraction) 2's-complement multiplier operating in the 10 to 20 ns range is considered. A high-speed multiplier of this type will find application in high-performance computers such as the Phoenix system.[5] The n-bit multiplicand (or multiplier) is partitioned into ℓ bytes of length (m - 1) for the least significant (ℓ - 1) bytes, and length b for the most significant byte, where $b \leq m$. For summand (partial product) generation, ℓ^2 mxm-bit multipliers are used. The optimum values for m are 9, 13, 17, and 21. The resultant ℓ^2 summands are then summed by one row of (p,2) counters to generate a partial-sum (PS) and a partial-carry (PC). The maximum value of p is related to ℓ by $p = 1+2(\ell - 1)$. The optimum choices for (p,2) counters are (3,2), (5,2), and (7,2). This multiplication process is illustrated in Fig. 1 (and a (7,2) counter is shown in Fig. 2).

In column summation using (p,2) counters, each counter compresses a column containing p bits by adding the column independently into C bits of carry and q = 2 bits of PS and PC. The C bits of carry are then added into the next higher order column with a maximum delay of two full adder stages. The final product is obtained by adding PS and PC in a standard carry-propagate adder (CPA) with fast carry look-ahead.

Values of m

Currently, mxm-bit multipliers are not available in the ECL technology, but multipliers with values of m equal to 8, 12, 16, and 24 are available in the STTL technology from two commercial sources.[6,7] For floating-point arithmetic, the above values of m are not optimum. A survey of the floating-point data format of several commercial computers is shown in Table 1. From this table, it can be seen that optimum values for m are 9, 13, 17, and 21, and that the signed-magnitude representation is more favorable. As an example for m = 13 as opposed to a naive choice of m = 12, the summand generation for Z = XY

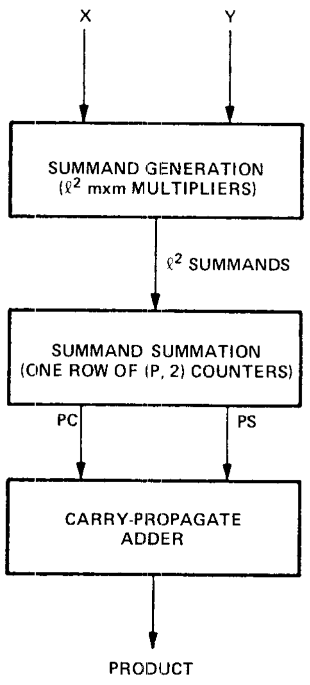
X
Y
SUMMAND GENERATION (ℓ^2 mxm MULTIPLIERS)
ℓ^2 SUMMANDS
SUMMAND SUMMATION (ONE ROW OF (P, 2) COUNTERS)
PC
PS
CARRY-PROPAGATE ADDER
PRODUCT

Figure 1. High-speed multiplication.

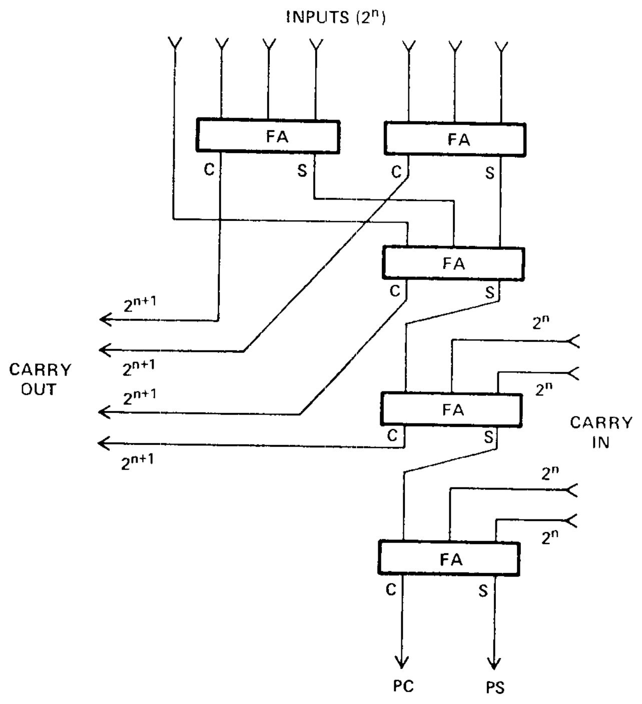

Figure 2. (7,2) counter.

Table 1. Floating-point format of several computers

Type	Single precision[a]				Double precision[a]				Number representation
	S	EXP	F	SD	S	EXP	F	SD	
Eclipse S/130	1	7	24	6-7	1	7	56	16	
PDP-11/70 (VAX11/780)	1	8	24	6-7	1	8	56	16	Signed-mag.
AP-120B	1	10	27	7-8					2's-compl.
PDP-10	1	8	27	7-8					2's-compl.
IBM system/360 and 370	1	7	24	6-7	1	7	56	16	Signed-mag.
ILLIAC IV	1	7	24	6-7	1	15	48	14	Signed-mag.
Burrough BSP	1	11	36	10-11	1	11	72	21-22	Signed-mag.
CDC 6600	1	11	48	14					1's-compl.
CDC STAR	1	8	23	6-7	1	16	47	13-14	2's-compl.
Cray 1	1	15	48	14					
Intel's μp	1	8	23+	6-7	1	11	52+	15	Signed-mag.

[a]S = sign, EXP = exponent, F = fraction, SD = significant digits.

where both X and Y are sign plus 24-bit fractions, regardless of the type of representation, the number of 12×12 multipliers required is nine whereas the number of 13×13 multipliers required is only four. In Table 1, Intel's proposed standard format is for microprocessors, and the fractional part has a hidden bit.[8]

mxm-Bit Multiplier

In terms of logic, 2's-complement multiplication is more complex than signed magnitude multiplication because the sign of the 2's-complement number is a value bit embedded in the number. But for a general-purpose multiplier chip, it is probably advantageous to manufacture 2's-complement, instead of signed-magnitude, mxm-bit multipliers. Currently, there are several algorithms for 2's-complement multiplications. Three of these are (1) Robertson's first method,[9] (2) modified Booth's method,[9,10] and (3) Baugh and Wooley method.[11] The modified Booth's method is probably the most popular one, and the mxm-bit multiplier can be implemented by using this method. Since this method is well-known, it will be reviewed only briefly here. In this method, the n-bit multiplier Y is recoded to a redundant number of higher radix, usually radix 4 or 5, without changing the natural value of Y. For both algorithms, the number of recoded multiples is $(n + 1)/2$ if n is odd and $n/2$ if n is even. It should be pointed out that if the natural value of n is odd, the result of radix-5 recoding is always one too high. This is because radix-5 recoding cannot produce the +1 or -1 multiples. For this reason, radix-4 recoding is generally more popular.

Summand Generation

The first step in high-speed multiplication is to generate the summands. The second step is to add the summands all at once to produce PC and PS. Finally, the product Z is obtained as

$$Z = XY = (PC + PS) \tag{1}$$

Let the multiplicand X and the multiplier Y each have n bits, a sign bit plus $n - 1$ bits of fraction, with negative values of X and Y represented in 2's-complement form. In order to use the mxm-bit multipliers for summand generation, it is necessary to partition X and Y into ℓ bytes as $X_{\ell-1}X_{\ell-2}X_{\ell-3},\ldots,X_1X_0$ and $Y_{\ell-1}Y_{\ell-2}Y_{\ell-3},\ldots,Y_1Y_0$, respectively. The $(\ell - 1)$ bytes of X_k and Y_k, $k = \ell-2,\ell-3,\ldots,1,0$, are chosen to have a byte length of $(m - 1)$ bits. When an mxm-bit multipler is used to obtain the product of any two of these bytes, the multipler input sign bits can be wired with positive signs so that the resultant product sign is always positive, and thus can be ignored. If the length of X_k and Y_k are $(m - 1)$, then the length of the byte $X_{\ell-1}$ is b where $b \le m$. For these two choices of byte length, the value of X and Y can be written as

$$X = \sum_{i=0}^{\ell-1} X_i 2^{(m-1)i} = X_{\ell-1} 2^{(m-1)(\ell-1)} + \sum_{k=0}^{\ell-2} X_k 2^{(m-1)k} \tag{2}$$

$$Y = \sum_{i=0}^{\ell-1} Y_i 2^{(m-1)i} = Y_{\ell-1} 2^{(m-1)(\ell-1)} + \sum_{k=0}^{\ell-2} Y_k 2^{(m-1)k} \tag{3}$$

where

$$0 \le X_{\ell-1} \ , \quad Y_{\ell-1} \le 2^{b-1} - 1 \tag{4}$$

and

$$0 \le X_k, Y_k \le 2^{m-1} - 1 \tag{5}$$

for $k = \ell-2,\ell-3,\ldots,1,0$. From Eqs. (2) and (3), the product Z is

$$Z = Z_3 + Z_2 + Z_1 + Z_0 \tag{6}$$

where

$$Z_3 = X_{\ell-1}Y_{\ell-1} \ 2^{2(m-1)(\ell-1)} \tag{7}$$

$$Z_2 = Y_{\ell-1} \sum_{k=0}^{\ell-2} X_k \ 2^{(m-1)(k+\ell-1)} \tag{8}$$

$$Z_1 = X_{\ell-1} \sum_{k=0}^{\ell-2} Y_k \ 2^{(m-1)(k+\ell-1)} \tag{9}$$

$$Z_0 = \sum_{i=0}^{\ell-2} \sum_{j=0}^{\ell-2} X_i Y_j \ 2^{2(m-1)k} \tag{10}$$

The number of summands in Eq. (6) is ℓ^2. For high-speed multiplication, $\ell \le 4$ should be the limit. This limits the number of summands to a maximum of 16.

The term Z_3 has only one summand. This summand has $[1 + 2(b - 1)]$ bits, a sign bit plus $2(b - 1)$ value bits where $b \le m$. The terms Z_2 and Z_1 each have $(\ell - 1)$ summands. Each summand consists of $[1 + (b - 1) + (m - 1)]$ bits, a sign bit plus $[(b - 1) + (m - 1)]$ value bits. When these summands are added with the Z_3 summand, their sign bits must be extended in order to produce the correct result. The term Z_0 has $(\ell - 1)^2$ summands. These summands have no sign because their sign bits are always positive. Therefore, extension of these sign bits is not required when these summands are added to Z_3; this will greatly simplify the summand summation, as described later. From the above discussion, the total number of summands contained in Z_1, Z_2, and Z_3 is $1 + 2(\ell - 1)$, and all of them contained a sign bit. When these summands are added all at once by (p,2) counters, the maximum value of p is therefore $p = 1 + 2(\ell - 1)$.

(p,2) Counter

Before proceeding into the discussion of summand summation, the design of the (p,2) column adder, called a (p,2) counter, is introduced first. In a traditional (p,q) counter, p input bits of equal weight are added to produce q output bits where $q = 1 +$ (integer part of $\log_2 p$), and the positional weight of the q bits are powers of 2. In the (p,2) counter, p input bits of equal weight are added to produce C bits of interstage carry and two output bits of partial sum (PS) and partial carry (PC). The C bits of carry are then added to the next higher-order stage with a delay of at most two full adders. Since the (p,2) counter has only two outputs, the summation of p n-bit summands can be performed by n + 1 (p,2) counters followed by a standard carry-propagate adder (CPA) with fast carry look-ahead.

The optimum values of p, as described in the next section, are 3, 5, and 7. The (3,2) counter is simply a full adder (FA). The logic diagrams of the (7,2) and the (5,2) counters are shown in Figs. 2 and 3, respectively.

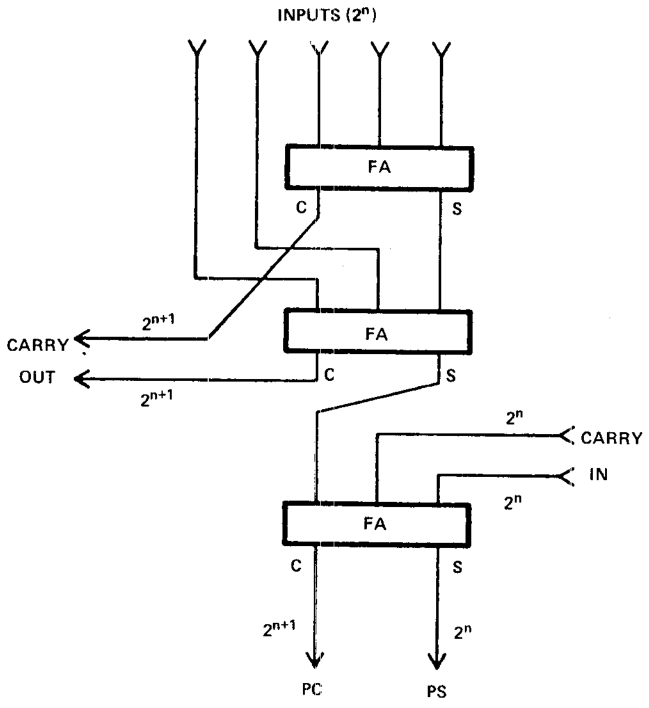

Figure 3. (5,2) counter.

<u>Summand Summation</u>

The summand generation logic uses ℓ^2 mxm-bit multipliers to generate ℓ^2 summands. These summands are skewed with respect to each group because of their positional weight differences. At any column, there are at most $p = 1 + 2(\ell - 1)$ summands to be added. The summation of these summands can be performed by using (p,2) counters to add each column independently and account for interstage carry bits. To illustrate this technique, consider the multiplication of X and Y, where X and Y are sign plus 24-bit fractions. If 13×13-bit multipliers are used, X and Y can be partitioned into two parts as follows:

$$\underset{X_1, Y_1}{(S\ \ 23\ \ 22\ \ 21 \text{——} 13\ \ 12)}\quad \underset{X_0, Y_0}{(11\ \ 10\ \ 9 \text{——} 1\ \ 0)}$$

The product $Z = XY$ has four summands, and is

$$Z = X_1Y_1\ 2^{24} + (X_0Y_1 + X_1Y_0)\ 2^{12} + X_0Y_0 \qquad (11)$$

This multiplication and summand summation using (3,2) counters are shown in Fig. 4. The small x used in Fig. 4 is a means of depicting the multiplication process and will be used throughout this discussion.

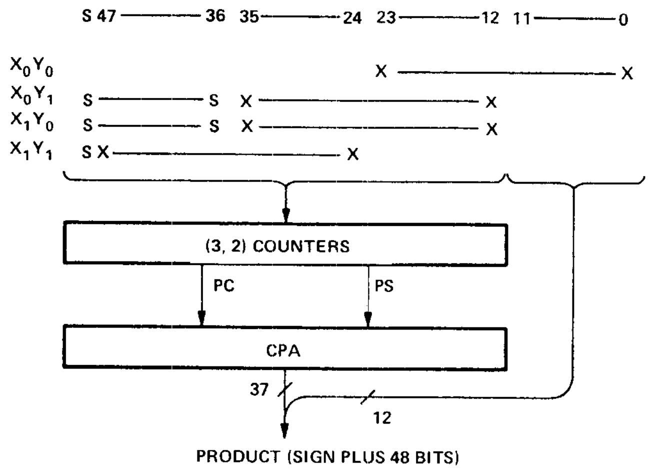
S 47 —— 36 35 —— 24 23 —— 12 11 —— 0
X_0Y_0
X_0Y_1
X_1Y_0
X_1Y_1
(3, 2) COUNTERS
PC
PS
CPA
37
12
PRODUCT (SIGN PLUS 48 BITS)

Figure 4. Multiplication using 13×13 multipliers.

If 9×9-bit multipliers are used, X and Y can be partitioned into three parts as 9, 8, and 8 bits. The product Z is

$$\begin{aligned} Z = {} & X_2Y_2\ 2^{32} + (X_1Y_2 + X_2Y_1)2^{24} \\ & + (X_0Y_2 + X_1Y_1 + X_2Y_0)2^{16} \\ & + (X_0Y_1 + X_1Y_0)2^8 + X_0Y_0 \end{aligned} \qquad (12)$$

This multiplication and summand summation using (5,2) and (3,2) counters are shown in Fig. 5. Similarly, the multiplication of two sign-plus-56-bit fractions using 17×17 and 9×9 multipliers has 16 summands. The 57 bits of X and Y are partitioned into four parts as 9, 16, 16, and 16 bits.

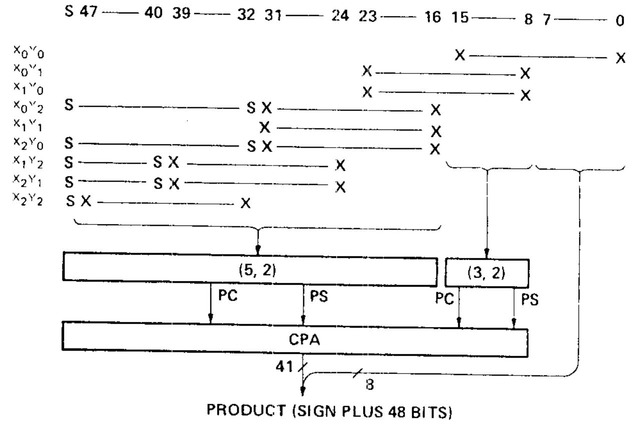

Figure 5. Multiplication using 9×9 multipliers.

This multiplication can also be summed in a similar manner by using (7,2), (5,2), and (3,2) counters. From the illustration in these figures, it can be seen that column summation requires (p,2) counters for values of p equal to 3,5, and 7, and higher values of p are not required.

Conclusion

A high-speed 2's-complement multiplication scheme has been described in this paper. The goal is to obtain at least a 32-bit floating-point (sign plus 24-bit fraction) multiplication in 10 to 20 ns using ECL LSI packages. The n-bit multiplicand (or multiplier) is partitioned into ℓ bytes of length $(m - 1)$ for the least significant $(\ell - 1)$ bytes, and length b for the most significant byte, where $b \leq m$. The multiplication uses ℓ^2 mxm-bit multipliers for summand generation and a row of (p,2) counters for summand summation. With respect to present computer floating-point formats, the optimum values for m are 9, 13, 17, and 21. For these values of m, the optimum values for p are 3, 5, and 7, and higher values of p are not required. The maximum value of p is related to ℓ by $p = 1 + 2(\ell - 1)$. With present technology, several (p,2) counters can be packaged into one ECL LSI chip.

For maximum throughput, the multiplication process can be pipelined into three sections as shown in Fig. 1. In this case, the mxm-bit multiplier and the (p,2) counter should have a register at their outputs for staging purposes. Although these multipliers and (p,2) counters currently are not available as ECL LSI packages, they can be manufactured if there exists a requirement. At this writing, Fairchild plans to introduce a (9,2) counter in an ECL SSI package called the F100182 with a delay of about 10 ns. If the multiplication is pipelined, then a 10-20 ns multiplication speed can be obtained.

In Fig. 4, there is no carry propagation in the row of (3,2) counters, while in Fig. 5, there is a carry propagation in the row of (5,2) counters. In Fig. 5, it is also possible to use (3,2) counters instead of (5,2) counters. If this is done, then an additional row of (3,2) counters is needed for bit positions 16 to s. This is undesirable because the objective here is to minimize the number of chips used.

Acknowledgment

The author gratefully acknowledges the help of his colleagues, David K. Stevenson and Gary F. Feierbach, in reading and commenting on the work reported herein.

References

1. Singh, S.; and Waxman, R.: Multiple Operand Addition and Multiplication. IEEE Trans. on Comp., Feb. 1973, vol. c-22, no. 2, pp. 113-120.

2. Ho, I. T.; and Chen, T. C.: Multiple Addition by Residue Threshold Functions and Their Representation by Array Logic. IEEE Trans. on Comp., Aug. 1973, vol. c-22, no. 8, pp. 762-767.

3. Agrawal, J. P.; and Reddy, V. U.: Log-Sum Multiplier. National Computer Conference, American Federation Information Processing Society, Inc., 1976, pp. 783-787.

4. Stenzel, W. J.; Kubitz, W. J.; and Garcia, G. H.: A Compact High-Speed Parallel Multiplication Scheme. IEEE Trans. on Comp., Oct. 1977, vol. c-26, no. 10, pp. 948-957.

5. Feierbach, G. F.; Stevenson, D. K.: A Prospectus for High-Speed Computing. IAC Phoenix Project Memo No. 006, Feb. 25, 1977. Institute for Advanced Computation, Sunnyvale, Calif. 94086.

6. TRW Inc., P.O. Box 1125, Redondo Beach, Calif. 90278. The mxm-bit multipliers are available for m equals to 8, 12, 16, and 24.

7. Monolithic Memories Inc., 1165 East Arques Ave., Sunnyvale, Calif. 94086. The 8×8-bit multiplier is the MM67558.

8. Palmer, J. F.: The Intel Standard for Floating-Point Arithmetic. ACM SIGNUM Meeting on Math. Software, Nov. 3-4, 1977, Albuquerque, New Mexico, pp. 107-112.

9. Chu, Y.: Digital Computer Design Fundamentals. McGraw-Hill, N.Y., 1962.

10. There is no unique reference and almost all high-speed computers used this method. For example, see Anderson, S. F., et al.: The IBM System/360 Model 91: Floating-Point Execution Unit. IBM Journ. of Res. and Dev., vol. II, no. 1, Jan. 1967, pp. 34-53.

11. Baugh, C. R.; Wooley, B. A.: A Two's Complement Parallel Array Multiplication Algorithm. IEEE Trans. on Comp., Dec. 1973, vol. c-22, no. 12, pp. 1045-1047.