A Two's Complement Parallel Array Multiplication Algorithm

CHARLES R. BAUGH, MEMBER, IEEE, AND BRUCE A. WOOLEY, MEMBER, IEEE

Abstract—**An algorithm for high-speed, two's complement, *m*-bit by *n*-bit parallel array multiplication is described. The two's complement multiplication is converted to an equivalent parallel array addition problem in which each partial product bit is the AND of a multiplier bit and a multiplicand bit, and the signs of all the partial product bits are positive.**

Index Terms—**Array multiplier, binary multiplication, high-performance multiplication, parallel multiplier, two's complement multiplication.**

MULTIPLICATION is often an essential function in digital systems. For example, it is a necessary operation in digital filtering and Fourier transform processing. With the advent of high-speed semiconductor memory, an increasing mismatch between memory access and multiplication time has arisen. Consequently, there exists considerable interest in parallel array multipliers. Most of this interest has centered around two's complement multipliers, since the two's complement representation of numbers is used almost universally. Two's complement representation adds complexity to the multiplication algorithm because the sign of the number is embedded in the number itself. This is in contrast to sign-magnitude representation where the sign and the number can be separated and multiplication is much simpler.

With the advent of medium-scale integration, the fabrication of parallel array multipliers has become economically more feasible. Several companies now offer modules that can be interconnected to form parallel array multipliers. For sign-magnitude multiplication, Fairchild offers a 2 × 4-bit multiplier [1], Hughes an 8 × 8-bit multiplier [2], and Texas Instruments a 4 × 4-bit multiplier [3]. For two's complement multiplication, a 2 × 4-bit multiplier is available from Advanced Micro Devices [4]. There have been at least two custom circuit designs for two's complement multipliers. Pezaris designed three circuits for implementing a 17 × 17-bit multiplier [5], while Hampel *et al.* used a single circuit for $n \times m$-bit multiplication [6]. Both designs were for high-speed multiplication. All commercial multipliers, except for the Advanced Micro Devices circuit, use conventional algorithms for implementing the multiplication.

In this paper an algorithm for parallel two's complement multiplication is described. The algorithm's principle advantage is that the signs of all the partial product bits are positive, allowing the product to be formed using array addition techniques. In conventional two's complement multiplication there are partial product bits with negative, as well as positive, signs.

In binary multiplication the $n + m$-bit product $P = (p_{n+m-1}, p_{n+m-2}, \cdots, p_0)$ is formed by multiplying the m-bit multiplicand $Y = (y_{m-1}, y_{m-2}, \cdots, y_0)$ by the n-bit multiplier $X = (x_{n-1}, \cdots, x_0)$. This multiplication is usually depicted as shown in Fig. 1. The AND of each multiplier bit and each multiplicand bit is formed to produce the partial product bits. The partial products are then summed to form the product.

The difficulty in two's complement multiplication lies with the signs of the multiplicand and the multiplier. Let Y_v be the value of the multiplicand Y, and X_v the value of the multiplier X. For two's complement representation X_v and Y_v are given by

$$Y_v = -y_{m-1} 2^{m-1} + \sum_{i=0}^{m-2} y_i 2^i \tag{1a}$$

$$X_v = -x_{n-1} 2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i. \tag{1b}$$

The value P_v of the product P is

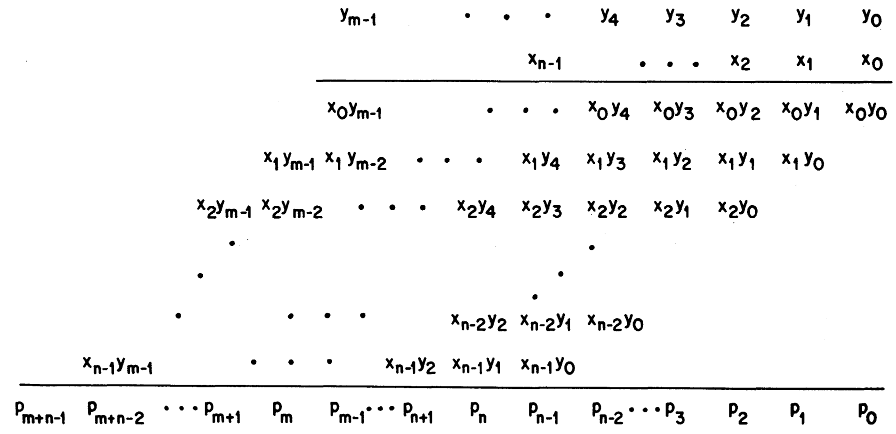
$$\begin{aligned} P_v &= -p_{m+n-1} 2^{m+n-1} + \sum_{i=0}^{m+n-2} p_i 2^i = Y_v X_v \\ &= \left(-y_{m-1} 2^{m-1} + \sum_{i=0}^{m-2} y_i 2^i\right)\left(-x_{n-1} 2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i\right) \\ &= \left(x_{n-1} y_{m-1} 2^{m+n-2} + \sum_{i=0}^{n-2} \sum_{j=0}^{m-2} x_i y_j 2^{i+j}\right) \\ &\quad - \left(\sum_{i=0}^{m-2} x_{n-1} y_i 2^{n-1+i} + \sum_{i=0}^{n-2} y_{m-1} x_i 2^{m-1+i}\right). \end{aligned} \tag{2}$$

When forming P by adding the partial products, the sign of the partial product bits must be considered. In particular, the signs of $x_{n-1} y_i$ for $i = 0, \cdots, m-2$ and $y_{m-1} x_i$ for $i = 0, \cdots, n-2$ are negative. By rewriting the partial product bits as shown in Fig. 2, all the partial product bits with negative signs are placed in the last two rows. The product is formed by adding the first $n - 2$ partial product rows and subtracting the last two rows.

Instead of subtracting the partial products that have negative signs, the negation of the partial products can be added. The

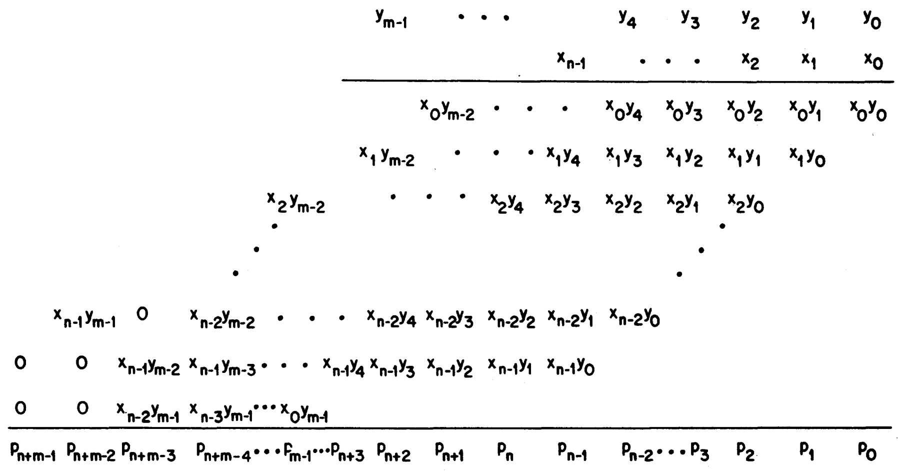
Manuscript received March 8, 1973; revised June 28, 1973. A summary of this paper was presented as a Short Note at COMPCON 73, the 7th Annual IEEE Computer Society International Conference, February 28, 1973.

The authors are with Bell Laboratories, Holmdel, N.J. 07733.



y_{m-1} • • • y_4 y_3 y_2 y_1 y_0
x_{n-1} • • • x_2 x_1 x_0
x_0y_{m-1} • • • x_0y_4 x_0y_3 x_0y_2 x_0y_1 x_0y_0
x_1y_{m-1} x_1y_{m-2} • • • x_1y_4 x_1y_3 x_1y_2 x_1y_1 x_1y_0
x_2y_{m-1} x_2y_{m-2} • • • x_2y_4 x_2y_3 x_2y_2 x_2y_1 x_2y_0
• • • x_{n-2}y_2 x_{n-2}y_1 x_{n-2}y_0
x_{n-1}y_{m-1} • • • x_{n-1}y_2 x_{n-1}y_1 x_{n-1}y_0
p_{m+n-1} p_{m+n-2} • • • p_{m+1} p_m p_{m-1} • • • p_{n+1} p_n p_{n-1} p_{n-2} • • • p_3 p_2 p_1 p_0

Fig. 1. Conventional two's complement $m \times n$-bit multiplication.



y_{m-1} • • • y_4 y_3 y_2 y_1 y_0
x_{n-1} • • • x_2 x_1 x_0
x_0y_{m-2} • • • x_0y_4 x_0y_3 x_0y_2 x_0y_1 x_0y_0
x_1y_{m-2} • • • x_1y_4 x_1y_3 x_1y_2 x_1y_1 x_1y_0
x_2y_{m-2} • • • x_2y_4 x_2y_3 x_2y_2 x_2y_1 x_2y_0
x_{n-1}y_{m-1} 0 x_{n-2}y_{m-2} • • • x_{n-2}y_4 x_{n-2}y_3 x_{n-2}y_2 x_{n-2}y_1 x_{n-2}y_0
0 0 x_{n-1}y_{m-2} x_{n-1}y_{m-3} • • • x_{n-1}y_4 x_{n-1}y_3 x_{n-1}y_2 x_{n-1}y_1 x_{n-1}y_0
0 0 x_{n-2}y_{m-1} x_{n-3}y_{m-1} • • • x_0y_{m-1}
p_{n+m-1} p_{n+m-2} p_{n+m-3} p_{n+m-4} • • • p_{m-1} • • • p_{n+3} p_{n+2} p_{n+1} p_n p_{n-1} p_{n-2} • • • p_3 p_2 p_1 p_0

Fig. 2. Positive/negative segregation of partial product bits.

value of the negation of a two's complement number $Z = (z_{k-1}, \cdots, z_0)$ with value Z_v is

$$-Z_v = 1 - \bar{z}_{k-1}2^{k-1} + \sum_{i=0}^{k-2} \bar{z}_i 2^i \tag{3}$$

where $\bar{z}_i$ is the complement of z_i. Therefore, the subtraction of

$$2^{n-1}\left(-0 \cdot 2^m + 0 \cdot 2^{m-1} + \sum_{i=0}^{m-2} x_{n-1}y_i 2^i\right) \tag{4}$$

can be replaced with the addition of

$$2^{n-1}\left(-1 \cdot 2^m + 1 \cdot 2^{m-1} + 1 + \sum_{i=0}^{m-2} \overline{x_{n-1}y_i} 2^i\right). \tag{5}$$

Thus the partial product row of Fig. 2 containing

$$0 \ 0 \ x_{n-1}y_{m-2} \ x_{n-1}y_{m-3} \ \cdots \ x_{n-1}y_0$$

is replaced by

$$1 \ 1 \ \overline{x_{n-1}y_{m-2}} \ \overline{x_{n-1}y_{m-3}} \ \cdots \ \overline{x_{n-1}y_0}$$

with a "1" added to the p_{n-1} column. Similarly, the row containing

$$0 \ 0 \ x_{n-2}y_{m-1} \ x_{n-3}y_{n-1} \ \cdots \ x_0y_{m-1}$$

is replaced by

$$1 \ 1 \ \overline{x_{n-2}y_{m-1}} \ \overline{x_{n-3}y_{m-1}} \ \cdots \ \overline{x_0y_{m-1}}$$

with a "1" added in the p_{m-1} column. Following these substitutions, all partial product bits can be treated in exactly the same manner with respect to the sign.

The substitution of (5) for (4) in Fig. 2 results in nonuniformity with regard to partial product bits since some partial product bits are the NAND of a multiplier bit and multiplicand bit, while others are formed with an AND. To simplify this situation, the following equivalences are used. Note that (5) has the value

$$\begin{cases} 0, & \text{for } x_{n-1} = 0 \\ 2^{n-1}\left(-2^m + 2^{m-1} + 1 + \sum_{i=0}^{m-2} \bar{y}_i 2^i\right), & \text{for } x_{n-1} = 1. \end{cases} \tag{6}$$

From (6) it follows that (5) can be rewritten as

$$2^{n-1}\left(-2^m + 2^{m-1} + \bar{x}_{n-1}2^{m-1} + x_{n-1} + \sum_{i=0}^{m-2} x_{n-1}\bar{y}_i 2^i\right). \tag{7}$$

Therefore, (5) is equivalent to (7). Since the next to the

$$
\begin{array}{cccccccccccccc}
 & & & & & y_{m-1} & \cdot & \cdot & \cdot & y_4 & y_3 & y_2 & y_1 & y_0 \\
 & & & & & & & & & x_{n-1} & \cdot\;\cdot\;\cdot & x_2 & x_1 & x_0 \\
\hline
 & & & & & & & x_0y_{m-2} & & \cdot\;\cdot & \cdot & x_0y_2 & x_0y_1 & x_0y_0 \\
 & & & & & x_1y_{m-2} & & \cdot & \cdot & \cdot & x_1y_2 & x_1y_1 & x_1y_0 & \\
 & & & & x_2y_{m-2} & & & \cdot & \cdot & \cdot & x_2y_1 & x_2y_0 & & \\
 & & & \vdots & & & & & & & \vdots & & & \\
x_{n-1}y_{m-1} & 0 & x_{n-2}y_{m-2} & & \cdot & \cdot & \cdot & & x_{n-2}y_2 & x_{n-2}y_1 & x_{n-2}y_0 & & & \\
\overline{x}_{n-1} & x_{n-1}\overline{y}_{m-2} & x_{n-1}\overline{y}_{m-3} & & \cdot & \cdot & \cdot & x_{n-1}\overline{y}_2 & x_{n-1}\overline{y}_1 & x_{n-1}\overline{y}_0 & & & & \\
\overline{y}_{m-1} & \overline{x}_{n-2}y_{m-1} & \overline{x}_{n-3}y_{m-1} & & \cdot\;\cdot\;\cdot & \overline{x}_0y_{m-1} & & & & & & & & \\
1 & & & & & y_{m-1} & & & & x_{n-1} & & & & \\
\hline
p_{n+m-1} & p_{n+m-2} & p_{n+m-3} & p_{n+m-4} & \cdots & p_{m-1} & \cdots & p_{n+1} & p_n & p_{n-1} & p_{n-2}\cdots p_3 & p_2 & p_1 & p_0
\end{array}
$$

Fig. 3. Algorithm with all positive partial product bits.

last row of partial product bits in Fig. 2 is of the form given in (5), (7) is substituted for this row. By making a similar substitution for the last row of partial product bits and by adding constants, the partial product bits of Fig. 3 are obtained.

The principle characteristic of the partial product bits in Fig. 3 is uniformity. The two advantages of this uniformity are as follows.

1) The partial product bits are obtained by forming the AND of a multiplier bit and a multiplicand bit.

2) Every partial product bit has a positive coefficient.

Therefore, the product is formed with only the AND function and the ADD function. No subtraction is necessary, nor is the NAND function needed to form $\overline{x_i y_j}$.

In a circuit implementation of the above algorithm, separate AND gates are not needed to form the *nm* partial product bits. Both Pezaris [5] and Hampel *et al.* [6] have designed circuits that readily incorporate the AND operation into the addition circuits. Also, the five extra partial product bits of the algorithm, x_{n-1}, $\bar{x}_{n-1}$, y_{m-1}, $\bar{y}_{m-1}$, and "1", can be included in a circuit implementation without increasing the total propagation delay of the two's complement multiplication.

A disadvantage of the proposed algorithm is the need for the complements of each multiplier and multiplicand bit in forming the partial product bits. However, for a number of reasons, this is not a major concern. Since high-speed multiplication is desired, current-mode (emitter-coupled) logic would most likely be used to implement the algorithm; with this logic both the output and its complement are available. In addition, the multiplier would probably be connected to a bus and driven from bus-receiving gates or a bus register; consequently, the complements of the multiplicand and multiplier bits would be readily available. Furthermore, since the fan-out requirements for the multiplicand and multiplier bits are high, special driving gates would be needed even if the multiplier was not connected to a bus; such driving gates would provide complemented signals.

References

[1] "TTL/MSI 9344 binary (4-bit by 2-bit) full multiplier," Fairchild Semiconductors, 313 Fairchild Drive, Mountain View, Calif.

[2] "Bipolar LSI 8-bit ,multiplier H1002MC," Hughes Aircraft Co., 500 Superior Ave., Newport Beach, Calif.

[3] "TTL/MSI SN74284-5 and SN54284-5 4-bit by 4-bit parallel binary multipliers," Texas Instruments, Inc., P.O. Box 5012, Dallas, Tex.

[4] "TTL/MSI AM2505 4-bit by 2-bit 2's complement multiplier," Advanced Micro Devices, 901 Thompson Place, Sunnyvale, Calif.

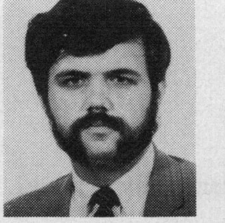
[5] S. D. Pezaris, "A 40-ns 17-bit by 17-bit array multiplier," *IEEE Trans. Comput.*, vol. C-20, pp. 442–447, Apr. 1971.

[6] D. Hampel, J. B. Lerch, and K. J. Prost, "Development of high-speed integrated circuits, digital multipliers and sample and hold gates; Final report," Air Force Avionics Lab., Contract AFAL-TR-71395, Mar. 1972.

Charles R. Baugh (S'65–M'70) received the B.S. degree in electrical engineering from Michigan State University, East Lansing, in 1965, the M.S. degree in electrical engineering, and the Ph.D. degree in computer science, both from the University of Illinois, Urbana, in 1967 and 1970, respectively.

He joined Bell Laboratories, Holmdel, N.J., in 1970. His current research interests are switching theory and digital signal processing.

Dr. Baugh is a member of Eta Kappa Nu and Sigma Xi.



Bruce A. Wooley (S'64–M'70) was born in Milwaukee, Wis., on October 14, 1943. He received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of California, Berkeley, in 1966, 1968, and 1970, respectively.

At the University of California he held appointments as Acting Instructor and Acting Assistant Professor in the Department of Electrical Engineering and Computer Sciences. He was employed by Motorola Semiconductor Products Division, Phoenix, Ariz., and Bell Laboratories, Allentown, Pa., during the summers of 1966 and 1968, respectively. Since 1970 he has been a Member of the Technical Staff at Bell Laboratories, Holmdel, N.J. His current research interests include the realization of functional integrated circuits for communication systems and the computer-aided design of integrated circuits.

Dr. Wooley is a member of Sigma Xi, Tau Beta Pi, and Eta Kappa Nu. In 1966 he received the University Medal from the University of California, Berkeley. He was the IEEE Fortescue Fellow for 1966–1967.