A TWO'S COMPLEMENT PIPELINE MULTIPLIER

Edmund K. Cheng*
California Institute of Technology, Pasadena, California 91125
Jet Propulsion Laboratory, Pasadena, California 91103

Carver A. Mead
California Institute of Technology, Pasadena, California 91125

SUMMARY

A serial-data pipeline multiplier was designed and implemented in p-channel silicon-gate MOS. It uses a radix-4 Booth algorithm for two's complement compatibility. The circuit is modular, and is configured to multiply one data word by two coefficient words simultaneously.

INTRODUCTION

The multiplier is often the most complicated arithmetic element in a digital signal processing system, and therefore it pays to find efficient ways of realizing this function. In fact, techniques using ROMs have been developed to avoid the use of multipliers entirely [1]-[2], highlighting the need for good multiplier design.

There have been several recent reports of multipliers that are fabricated in a variety of large-scale integration technologies for different levels of performance [3]-[6]. This paper will begin by pointing out some of the features in multiplier designs that are considered to be desirable for certain digital signal processing applications, and discuss a particular design that embodies these features.

THE PIPELINE MULTIPLIER

Multipliers can be divided into two major catagories, array multipliers and clocked or serial-data multipliers. The array multiplier uses extensive parallelism so that the final product is obtained without registering the partial products. It can therefore form the product faster than the serial-data multiplier. However, due to other considerations such as complexities, pin count, testability, etc, serial-data multipliers are often chosen for digital signal processors.

*Presently with Intel Corp., Santa Clara, CA 95051.

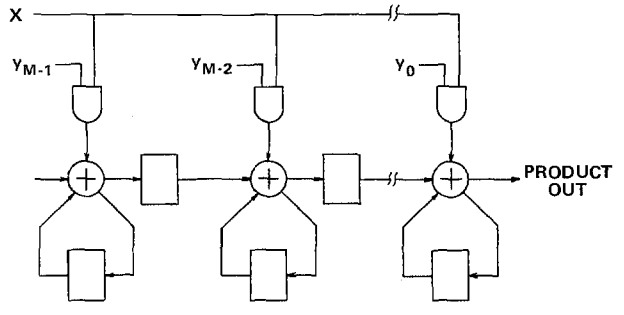

Fig. 1. Serial-parallel multiplier.

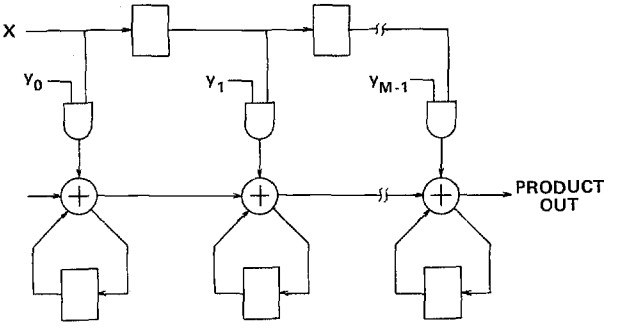

Fig. 2. Variation of Fig. 1.

The conventional serial-data multiplier is designed to simulate the pencil and paper method of binary multiplication, or the "shift-and-add gated multiplicand" algorithm. Fig. 1 shows a straightforward implementation of such a multiplier [4]-[7], where X is the multiplicand word, and Y is the multiplier word. This is a serial-parallel multiplier, and since all of the bits of Y must be assembled in parallel before multiplication can proceed, it is an inconvenience in certain serial-data systems.

Fig. 2 shows a different circuit configuration which can accommodate certain features that are desirable in serial-data digital signal processing systems, and they are described as follows.

In order to closely pack the serial data, successive words are adjacent in time. Therefore, no more than N bits per word are allowed at any point in the serial digital network. However, multiplying an N-bit multiplicand word by an M-bit multiplier word yields an (N+M)-bit product word. Since the product word is the output data word, it should be reduced to the same length as the input data word.

As a partial product passes through a multiplier section, it can grow in length by one bit via a carry-out of its high-order bit. The extra bit can be mistakenly added into the LSB of the succeeding partial product. Since it is preferable to preserve the high-order bits, the obvious solution is to truncate the entering partial product by setting its LSB to "0", thereby maintaining the length of a developing partial product to N bits throughout the multiplication.

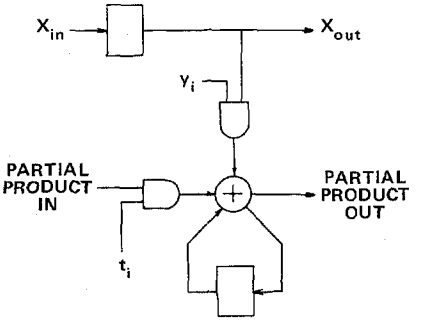

Fig. 3. Pipeline multiplier bit section with truncation.

Fig. 3 shows a single multiplier bit section of the pipeline multiplier [8] which is designed to accept positive data words that are closely packed in time and form products that are truncated (or rounded) to the same length, at the same rate. Truncation is accomplished by setting the truncation or timing signal T to "0" at the clock period when a partial product LSB is about to enter the bit section in question. The lower-order bits are removed as they are generated, but the carries due to them are preserved.

Since all the adders in the partial-product chain are connected in series, the progation delay is excessive, thereby slowing down the data rate. For high data rates, an extra set of delays is inserted between the multiplier bit sections to resynchronize the partial product after each addition. This technique doubles the number of bit cycles in going through the multiplier, which is considered to be inconsequential in many digital processors.

It should be clear that, by using appropriate latches, Y can be loaded one bit at a time, LSB first, simultaneously with X. The timing signal can be used as the latch-enable signals. It is also possible to vary Y from one multiplicand word to the next.

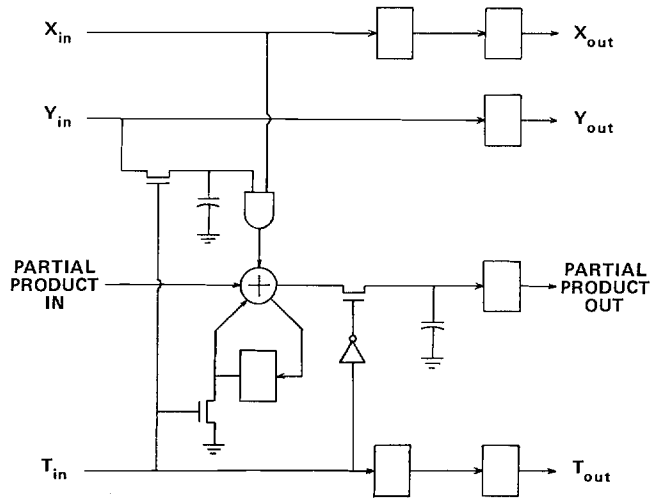

Fig. 4. Souped-up version of Fig. 3 in dynamic MOS.

To correctly process the multiplicand word as a two's complement number, sign extensions should be used when the word is lengthened at the most significant end. Two's complement addition can cause output overflow; in order that such an overflow does not interfere with the succeeding addition, the carry input of each adder should be cleared between multiplicand words. These functions can also be performed using the timing signal.

Fig. 4 illustrates a multiplier bit section that embodies all of the above modifications to that shown in Fig. 3; it is designed for realization in dynamic MOS.

THE TWO'S COMPLEMENT MULTIPLIER

Assuming that the binary point is to the right of the LSB, an M-bit two's complement number can be represented as:

$$Y = -y_{M-1}2^{M-1} + \sum_{i=0}^{M-2} y_i 2^i \qquad (1)$$

where each y_i is a bit in the number Y, and y_{M-1} is the sign bit. Consequently, in order to process the multiplier word as a two's complement number, the adder in the last multiplier stage should be replaced by a subtractor. Although this does not increase the amount of hardware significantly, it does destroy the modularity of the system, and makes it difficult to append more stages to accommodate longer multiplier words. The LSI implementation of such a multiplier would be greatly enhanced by modularity, and its application becomes much more flexible if each unit is expandable.

y_i	y_{i-1}	Operation
0	0	PP + 0
0	1	PP + X
1	0	PP - X
1	1	PP - 0

Table 1. Table of operations for the basic Booth algorithm.

The Booth algorithm [9] can be used in constructing a two's complement pipeline multiplier with a completely modular configuration. The basic Booth algorithm is outlined as follows. From Eq. 1,

$$Y = -y_{M-1}2^{M-1} - \sum_{i=0}^{M-2} y_i 2^i + 2\sum_{j=0}^{M-2} y_j 2^j$$

$$= -\sum_{i=0}^{M-1} y_i 2^i + \sum_{j=0}^{M-2} y_j 2^{j+1}$$

since

$$\sum_{j=0}^{M-2} y_j 2^{j+1} = \sum_{k=0}^{M-1} y_{k-1} 2^k$$

where $y_{-1}=0$, one can write

$$Y = -\sum_{i=0}^{M-1} y_i 2^i + \sum_{k=0}^{M-1} y_{k-1} 2^k$$

$$= \sum_{i=0}^{M-1} (y_{i-1} - y_i) 2^i \qquad (2)$$

Therefore, instead of multiplying X by y_i, add to the partial product, shift X, and repeat, one can multiply X by $(y_{i-1}-y_i)$ and so on. Table 1 shows the table of operations. Each multiplier bit y_i is examined together with its preceding bit y_{i-1}. PP is the partial product which is set to zero initially, and y_{-1} is defined to be zero.

To extend the algorithm one step further [10], two multiplier bits y_{i+1} and y_i are simultaneously examined with y_{i-1}, and the table of operations for this radix-4 Booth algorithm is shown in Table 2. The hardware must now be able to perform addition and subtraction of X as well as 2X. The 2X word can easily be obtained by shifting the X word one bit to the left. In order for it not to interfere with the next multiplicand word, a one bit time slot must be inserted between words. This time slot serves as the LSB for a 2X word, and should be filled with a "0"; however, it is also the MSB for an X word, and should be filled with a sign extension. These conditions are simultaneously satisfied by inserting a sign extension to the multiplicand word at the input of the multiplier, and inserting zeros within the multiplier stages that are performing the 2X operation. As a result, each multiplicand word consists of a sign bit, (N-2) magnitude bits plus a "don't care" bit.

The pipeline multiplier that was discussed earlier requires as many sections or stages as there are bits in the multiplier word. This method requires only half as many stages, without doubling the hardware in each stage.

			Operation		
Y_{i+1}	Y_i	Y_{i-1}	$Y_{i+1}Y_i$	Y_iY_{i-1}	Net Operation
0	0	0	PP + 0	PP + 0	PP + 0
0	0	1	PP + 0	PP + X	PP + X
0	1	0	PP + 2X	PP - X	PP + X
0	1	1	PP + 2X	PP - 0	PP + 2X
1	0	0	PP - 2X	PP + 0	PP - 2X
1	0	1	PP - 2X	PP + X	PP - X
1	1	0	PP - 0	PP - X	PP - X
1	1	1	PP - 0	PP - 0	PP - 0

Table 2. Table of operations for the radix-4 Booth algorithm.

Although this method can be extended to three or more bits, it will be more trouble than it is worth. For example, if three multiplier bits are simultaneously examined with their preceding bit, then the 3X word needs to be generated (it is no longer a simple shifting operation), and at least two adder-subtractors must be used per stage. Therefore, the radix-4 algorithm seems to be optimal for the application.

Fig. 5 shows the functional block diagram of a pipeline multiplier that uses the radix-4 Booth algorithm. Since a sizable amount of combinatorial logic is required in implementing Table 2, the multiplier word is examined by a decoder which then generates the necessary control signals for all the multiplier stages that follow. Such a configuration amortizes the decoding logic over the entire pipeline multiplier, making it a minor overhead.

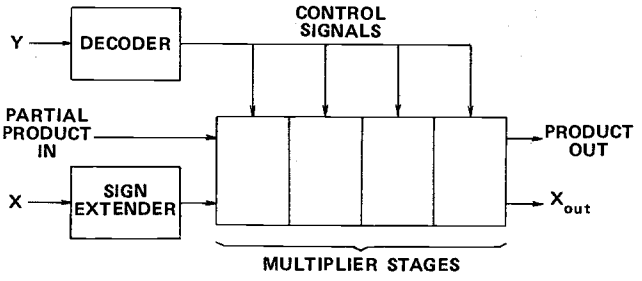

Fig. 5. Block diagram of two's complement pipeline multiplier.

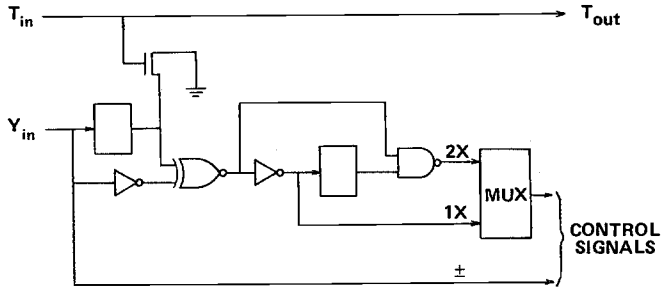

Fig. 6. Simplified logic diagram for the multiplier decoder.

THE DESIGN

A simplified logic diagram for the multiplier decoder is given in Fig. 6. Since the decoder examines two multiplier bits simultaneously, it has two bit cycles for sending out the control signals that it generates, and they are summarized in Table 3. The multiplier stages simply latch in these time-muliplexed control signals at the appropriate times, and process the multiplicand word accordingly. Fig. 7 shows a functional diagram for such a multiplier stage; it is simply a radix-4 version of that shown in Fig. 3, with the added capabilities for subtraction and two times the multiplicand.

In several applications for which this pipeline multiplier is designed, such as digital filters and complex multiplication in FFT processors, a data word is simultaneously multiplied by two coefficients, and this fact can be used to realize additional savings in hardware by constructing two multipliers as one unit and eliminating the unnecessary redundancies. The double pipeline multiplier accepts one multiplicand word (X) with arbitrary length and two multiplier words (a, b) with length (M bits) limited by the number of stages. Fig. 8 shows a photomicrograph of such an experimental device. Using very conservative design rules, this 8XN bits double multiplier is implemented in silicon gate PMOS, and measures under 50X100 mils in chip area.

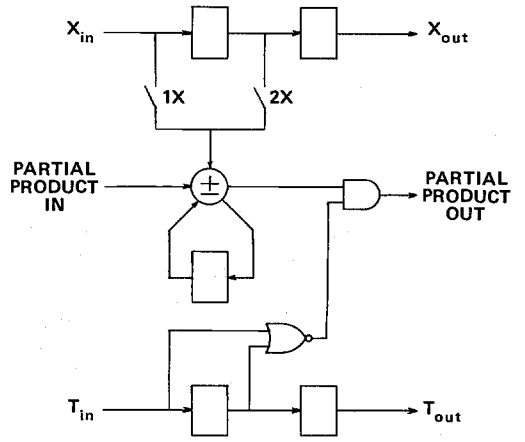

Fig. 7. Functional diagram of radix-4 multiplier stage.

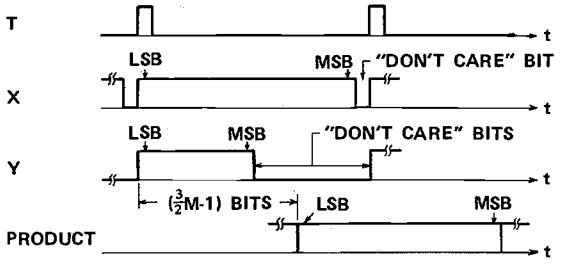

Fig. 9. Timing diagram.

Within each multiplier stage, two-bit truncation and double sign-extension are performed using the timing signal. A set of extra delays is used in each stage to resynchronize the partial product signals, but only increasing the amount of delay in the output product by 50%. As a result, the throughput (or clocking) rate is independent of the number of stages, and there is no fan-out problem to limit the number of stages. Dynamic shift registers and latches are used throughout to reduce chip area.

Fig. 9 shows the timing diagram for all the signals involved. The device has been tested at 2.9MHz bit rate.

APPLICATIONS

The two's complement pipeline multiplier has a delay of (3/2)M-1 bits in generating the product word. When used in a digital filter circuit, this delay must be incorporated into a delay (z^{-1}) that precedes the multiplier. If the data words are closely packed in time, z^{-1}=N. Therefore, the condition

$$(3/2)M-1 \leqslant N \qquad (3)$$

must be satisfied. In a typical digital filter, data word length N ranges from 12 to 20 bits, while coefficient word length M is 8 to 12 bits; obviously, Eq. (3) can be satisfied in most practical applications. A straightforward second-order digital filter implementation using this multiplier circuit is shown in Fig. 10, where the length of the delays D_1-D_8 are dependent upon the choice of N and M.

It has been found that when two's complement arithmetic is used, overflow oscillations can occur in the feedback loop of second-order filter sections with certain coefficient values [11]. If such instabilities are not considered a threat, then this simple structure can be used as a high speed programmable filter without any multiplexing. Results from the experimental multiplier chip indicate that such a second-order filter section would occupy about 100X110 mils in chip area, using silicon gate PMOS.

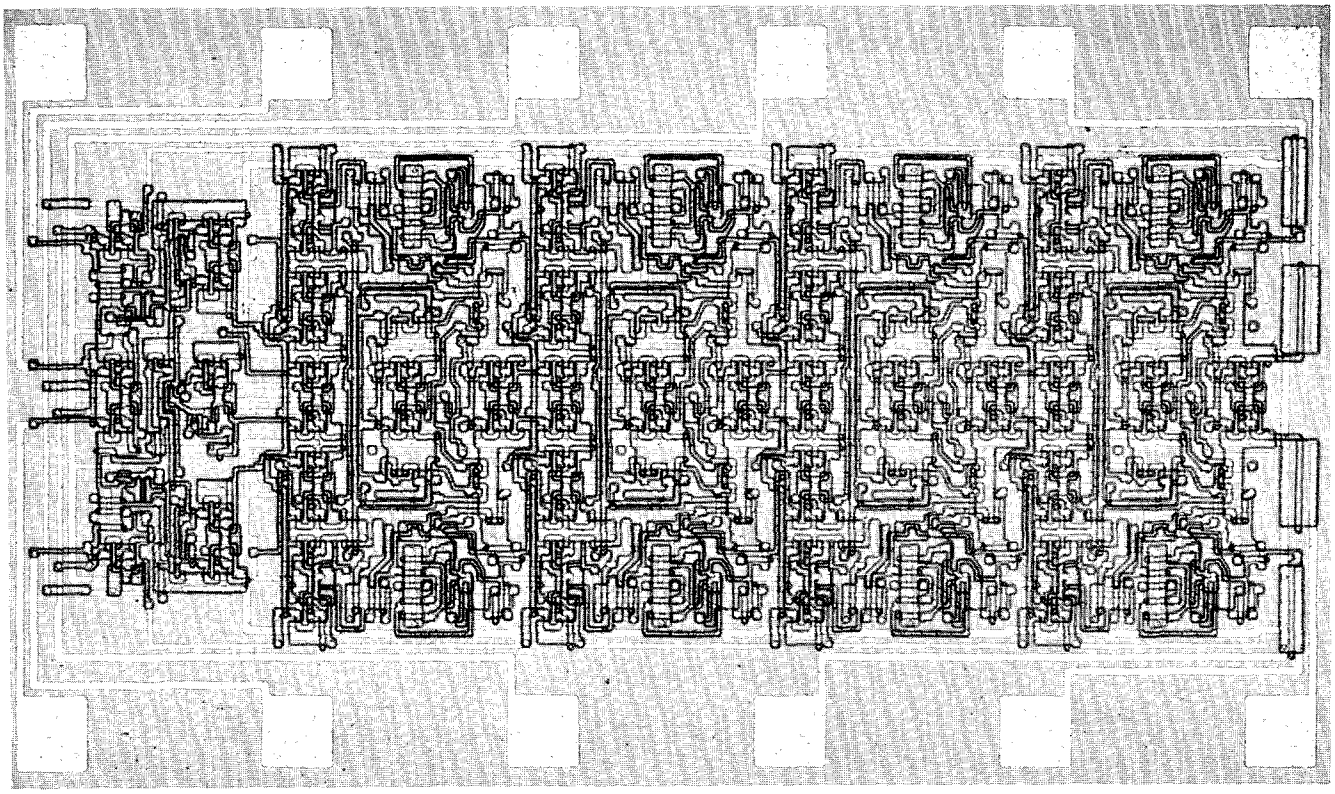

Fig. 8. Photomicrograph of 8XN bits double-multiplier.

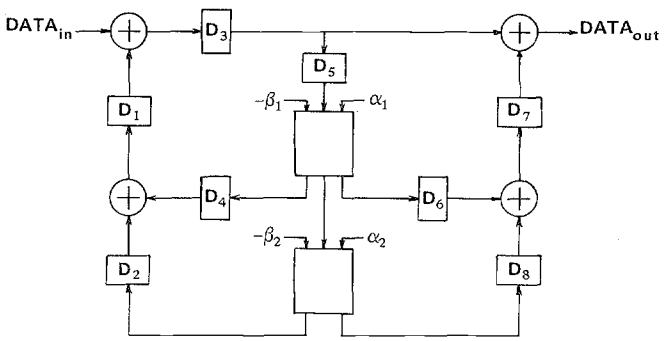

Fig. 10. Second-order section using double multiplier.

Two of these double multipliers can also be used in making a complex multiplier. A pipeline FFT processor section [12] that consists of a complex multiplier, an adder, a subtractor, and some switching logic can be implemented in a single LSI chip that fits in a small DIP package. When provided with appropriate shift register memories and coefficient ROMs, a straightforward pipeline FFT processor can be made with only a few of this LSI chip.

	1st bit-cycle	2nd bit-cycle
1X	0 1X	irrelevant
	1 no op	"
2X	irrelevant	0 2X
	"	1 no op
±	"	0 addition
	"	1 subtraction

Table 3. Control signals for the multiplier stages.

ACKNOWLEDGEMENTS

The authors wish to thank R. F. Jurgens, R. T. Masumoto, and G. A. Morris for their assistance; Intel Corp. for fabrication of the circuits; and in particular R. F. Lyon for his invaluable contributions during the initial phase of this work.

REFERENCES

[1] A. Peled and B. Liu, "A new hardware realization of digital filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-22, pp. 456-461, Dec. 1974.

[2] B. Liu and A. Peled, "A new hardware realization of high-speed fast Fourier transformers," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-23, pp. 543-547, Dec. 1975.

[3] G. W. McIver, R. W. Miller, and T. G. O'Shaughnessy, "A monolithic 16x16 digital multiplier," in *1974 Int. Solid-State Circuits Conf., Dig. Tech. Papers*, pp. 54-55.

[4] S. A. White and T. Mitsutomi, "The IC digital filter: a new low-cost signal processing tool," *Control Engineering*, vol. 43, pp. 58-68, June 1970.

[5] G. P. Edwards, P. J. Jennings, and T. Preston, "A MOS LSI double second order digital filter circuit," in *1975 Int. Solid-State Circuits Conf., Dig. Tech. Papers*, pp. 20-21.

[6] D. Hampel, K. E. McGuire, and K. J. Prost, "CMOS/SOS serial-parallel multiplier," *IEEE J. Solid-State Circuits*, vol. SC-10, pp. 307-314, Oct. 1975.

[7] R. K. Richards, *Arithmetic Operations in Digital Computers*. Princeton: Van Nostrand, 1955, p. 155.

[8] L. B. Jackson, J. F. Kaiser, and H. S. McDonald, "An approach to the implementation of digital filters," *IEEE Trans. Audio Electroacoust.*, vol. AU-16, pp. 413-421, Sept. 1968.

[9] A. D. Booth, "A signed binary multiplication technique," *Quart. J. Mech. Appl. Math.*, vol. 4, part 2, 1951.

[10] O. L. MacSorley, "High-speed arithmetic in binary computers," *Proc. IRE*, vol. 49, pp. 67-91, Jan. 1961.

[11] P. M. Ebert, J. E. Mazo, and M. G. Taylor, "Overflow oscillations in digital filters," *Bell Syst. Tech. J.*, vol. 48, pp. 2999-3020, Nov. 1969.

[12] H. L. Groginsky and G. A. Works, "A pipeline fast Fourier transform," *IEEE Trans. Comput.*, vol. C-19, pp. 1015-1019, Nov. 1970.