Correspondence

A Fast Serial–Parallel Binary Multiplier

R. GNANASEKARAN

Abstract — **A fast serial–parallel (FSP) multiplier design is derived from the carry–save add–shift (CSAS) multiplier structure. The CSAS technique accepts multiplier bits serially (lsb first) and produces outputs serially (lsb first). Multiplication of two n bit unsigned numbers requires $2n$ clock cycles to complete the process out of which n clocks are used for n-row carry–save additions, and the other n clocks are utilized only to propagate the remaining carries. This CSAS structure is modified so that it operates as a CSAS unit for the first n clocks and reconfigures itself as an n bit ripple-carry parallel adder at the $(n + 1)$st clock, thus allowing the carries to ripple through, eliminating the delay due to storage elements during the last n clocks. It is shown that this modification results in an about one-third increase in speed for an approximately one-third increase in hardware. The technique is extended to signed numbers represented in 2's complement form. Also, it is shown how these implementations can be modularized.**

Index Terms — **Add–shift multiplier, array multiplier, carry–save addition, pipeline multiplier, ripple-carry parallel adder, serial–parallel multiplier, 2's complement multiplication.**

I. Introduction

Almost all signal processing applications demand a considerable number of multiplications. Some of these applications require the multiplication to be performed at a faster rate, and others concentrate on less hardware and moderate speed.

In order to meet the demand for high speed, various parallel array multiplication algorithms have been proposed by a number of authors [1]–[8]. The array multipliers use a large amount of hardware, consequently consuming a large amount of power.

The applications which aim at hardware simplicity traditionally use, in one form or another, the add–shift method [9], [10] for multiplication. Referring to Fig. 1, the basic add–shift with the ripple-carry propagation technique adds, one row at a time, the multiplication matrix to the partial product using an n bit adder. In every addition, the carry is allowed to propagate full length. This carry propagation, along with n-row addition time, contributes largely to the slow speed. The speed can be improved by using the carry lookahead technique, which introduces more hardware in the implementation.

A nonconventional quasi-serial multiplier was introduced by Swartzlander [11], [12] in which the matrix in Fig. 1 is added by counting the number of ones in the individual columns. For n bit multiplication, it takes $2n - 1$ clock cycles, clock period being equal to the sum of the gate delay, the counter delay, the addition delay of the carry register, and the shift delay [11]. Chen and Willoner [13] recently forwarded a parallel multiplier with bit-sequential input and output for positive numbers, which requires $2n$ modules of (5, 3) counters. In [14] this technique was extended to 2's complement numbers, and also, it was shown that the multipliers can be implemented with only n modules of (5, 3) counters. The computation time is $2n$ clock pulses, and the clock period is one counter delay and the intermediate carry storage setup and settling times. Jackson *et al.* [15], [16] forwarded a serial–parallel pipelined multiplier structure especially suitable for digital signal processing applications where the data are available in series. This unit produces only most significant n bits of the product serially. High throughput rate can be obtained in a fully pipelined environment. Although it takes only n clocks to produce n most significant bits of the product in a pipelined environment, for single multiplication it takes $2n$ clocks. Moreover, it is not a preferable implementation where a full-length product is desired. It can be seen that the speed of all the above multipliers is comparable to that of a CSAS unit.

Manuscript received February 28, 1983; revised October 2, 1984.

The author is with the Department of Electrical Engineering and Computer Science, University of Nevada, Reno, NV 89557.

				a_4	a_3	a_2	a_1
				b_4	b_3	b_2	b_1
				$a_4 b_1$	$a_3 b_1$	$a_2 b_1$	$a_1 b_1$
			$a_4 b_2$	$a_3 b_2$	$a_2 b_2$	$a_1 b_2$	
		$a_4 b_3$	$a_3 b_3$	$a_2 b_3$	$a_1 b_3$		
	$a_4 b_4$	$a_3 b_4$	$a_2 b_4$	$a_1 b_4$			
P_8	P_7	P_6	P_5	P_4	P_3	P_2	P_1

Fig. 1. The product matrix.

This correspondence presents a fast serial–parallel (FSP) multiplier implementation which has an approximately one-third increase in hardware for an approximately one-third increase in speed compared to that of the conventional add–shift technique. First, positive numbers are considered. Then the procedure is extended to the signed numbers represented in 2's complement form. All the multiplicand bits are assumed to be available at the beginning. The realization does not need any prior knowledge of the signs of the operands.

II. FSP Multiplier for Unsigned Operands

A. Proposed Structure

A simple CSAS multiplier implementation for positive numbers is shown in Fig. 2. For n bit multiplication it takes $2n$ clock pulses to complete the process. At the end of the first n clocks the structure contains a sum word $\{S_n(n), S_{n-1}(n), \cdots, S_2(n)\}$ and a carry word $\{C_{n-1}(n), C_{n-2}(n), \cdots, C_1(n)\}$ the sum of which will yield n msb's of the product. This sum is accomplished by the next n clocks. Therefore, out of $2n$ clocks the last n clocks are expended only to propagate the carries residing in the feedback latches.

The multiplication time of the CSAS unit can be significantly improved if the addition of sum and carry words is performed with an $(n - 1)$ bit parallel adder instead of using n more clocks to propagate the carries. This will eliminate the accumulated delay due to storage elements during the last n clock intervals. (It is to be noted here that for a given circuit technology the speed advantage offered by a parallel array multiplier over the CSAS structure is primarily due to the total elimination of the intermediate storage and associated gate delays at the expense of a considerable increase in hardware. The implementation presented here exploits this tradeoff that exists between the hardware/power versus the speed for a given circuit technology.) To achieve this we propose here to modify Fig. 2 so that it operates as a CSAS unit for the first n clocks, and at the end of the n clock period it reconfigures itself as an $(n - 1)$ bit parallel adder circuit. This modification is shown in Fig. 3.

During the first $(n - 1)$ clock pulses, Q is zero and the structure acts like a CSAS unit. The outputs from the second sets of adders are ignored. The first $(n - 1)$ lsb's are shifted out serially at the output. Q is set to 1 by the nth clock, which in turn stops the system

0018-9340/85/0800-0741$01.00

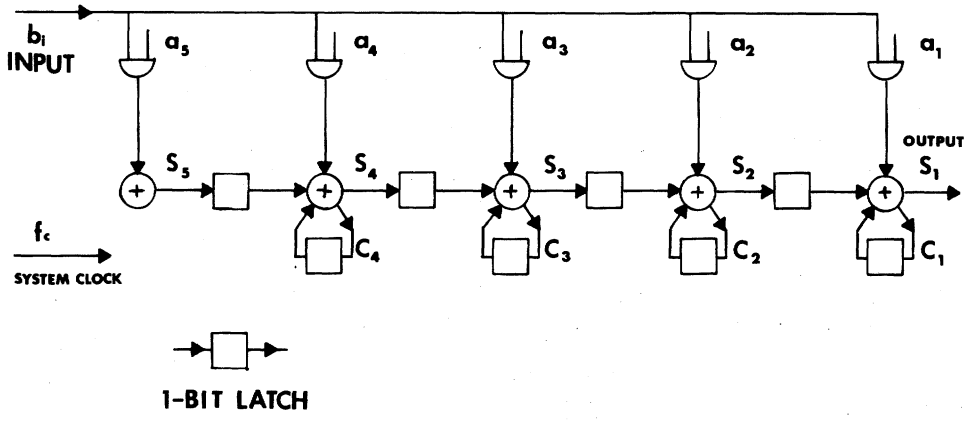

Fig. 2. Unsigned CSAS multiplier.

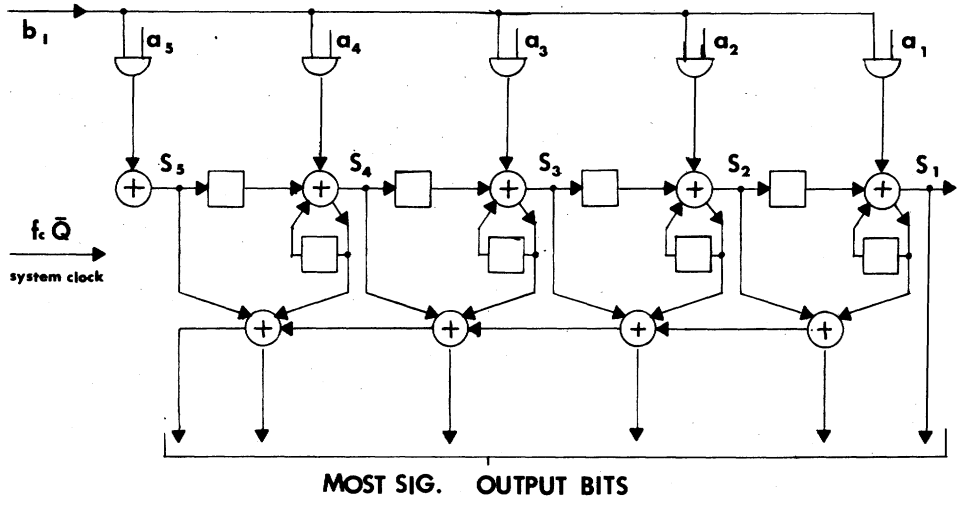

Fig. 3. Unsigned FSP multiplier.

clock. At this time the second set of adders performs the addition of the sum and carry words shown in Fig. 2.

B. Speed

During CSAS operation the clock interval is given by

$$T_C = t_{\text{latch}} + t_{\text{setup}} + t_{\text{AND}} + t_{\text{FA}} \tag{1}$$

where t_{latch} is the delay in the latch, t_{setup} is the setup time required by the latch, t_{AND} is the AND gate delay, and t_{FA} is the fuller adder delay. Note that the t_{latch} is due primarily to the multiplicand shift and the t_{setup} is due to the feedback latches. $(n - 1)$ bit parallel adder delay is

$$T_p = (n - 1)t_{\text{FA}}. \tag{2}$$

Therefore, the FSP multiplier delay is

$$T_{\text{FSP}} = nT_C + (n - 1)t_{\text{FA}}. \tag{3}$$

CSAS multiplier delay can easily be seen to be

$$T_{\text{CSAS}} = 2nT_C. \tag{4}$$

Therefore, the improvement in speed of the FSP multiplier over the CSAS multiplier is

$$T_{\text{gain}} = T_{\text{CSAS}} - T_{\text{FSP}} = n[t_{\text{latch}} + t_{\text{setup}} + t_{\text{AND}}] + t_{\text{FA}}. \tag{5}$$

It is seen that the speed improvement is mainly due to the elimination of the accumulated delay contributed by the storage elements and the AND gates during the later n clocks.

In [17] it is determined that the delay of the 1 bit full adder circuit together with the storage and gating elements is equivalent to

$$T_{\text{cell}} = \beta t_{\text{FA}} \tag{6}$$

where β is typically between 2 and 6. Observing that T_C is in fact equal to T_{cell}, and assuming a typical value for β to be 4 [17], we obtain

$$T_{\text{FSP}}/T_{\text{CSAS}} = 0.625. \tag{7}$$

There is an about one-third increase in speed for an approximately one-third increase in hardware. The speed comparison between FSP, CSAS, parallel array [17], and Chen and Willoner multipliers is shown in Table I. Here $\beta = 4$ is assumed for FSP, CSAS, and Chen and Willoner multipliers.

C. Hardware

In [17] it has been noted that the equivalent number of basic 1 bit full adder circuits for a single bit full adder and the associated delay and gate circuits of the serial multiplier in [15] is between $2 < \sigma < 5$, the typical value being $\sigma = 3$. Taking this as the model, each single bit circuit in the CSAS network is comparable to that of the model. Each single bit circuit in FSA contains one more full adder than that of CSAS. Each single bit circuit of Chen and Willoner's contains approximately two more full adders than that of CSAS, one due to the (5, 3) counter and the second due to the other gates and storage elements involved in the generation of the inputs to these counters [14]. The hardware comparison is shown in Table I. However, the hardware necessary for the modulo n counter

TABLE I
COMPARISON OF SPEED AND HARDWARE

	CSAS	FSP	Chen & Willoner	Parallel array
Hardware	3(n-1)	4(n-1)	5n	n(n-1)
Speed	8n t	(5n-1)t	8n t	n t

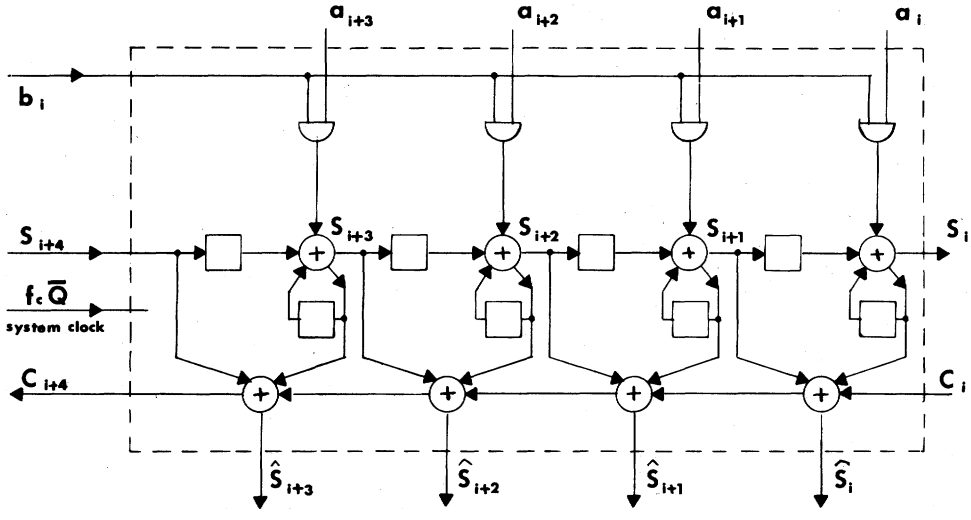

Fig. 4. 4 bit module of unsigned FSP multiplier.

used in FSP for the generation of Q, modulo $2n$ counters for CSAS, and Chen and Willoner units are not included in the comparison.

In the FSP multiplier, the lsb $(n-1)$ bits of the product are available in serial and the msb $(n+1)$ bits are available in parallel. If $n1$ is the number of clock intervals that cover the parallel adder delay, then the msb's can be clocked into a register at the end of $(n+n1)$ clocks using the same clock source. As an approximate calculation, from (2) and (6), with a typical value of $\beta = 4$, the number of clock pulses that cover the parallel adder delay is $(n-1)/4$. This suggests that the number of clocks that are saved by the FSP structure is about $(3/4)n$. Note that all the serial–parallel multipliers need a modulo counter to indicate the end of the multiplication. However, the FSP unit needs two modulo counters, one to generate Q and the other to store the msb's of the product.

The FSP multiplier can easily be modularized. As an example, a 4 bit module is shown in Fig. 4. A multiplier of any length can be constructed using these modules with proper interconnections.

III. 2's Complement Multiplier

A step-by-step add–shift multiplication algorithm for 6 bit 2's complement numbers is shown in Fig. 5. An important aspect to be observed here is that the sign bit extension can be either a direct extension of the sign bit of the partial product sum (PPS) (PPS's 1 and 4) or created as a result of a carry generated by previous row addition (PPS's 2, 3, and 5). Also, during the last addition, if b_n is 1, then 2's complement of the multiplicand is added; otherwise, zero is added.

Fig. 6 shows the FSP multiplier implementation for the 2's complement numbers. At adder 5, $Sn1$ and $Sn2$ correspond to the direct and due-to-the-carry sign bit extensions, respectively. $Q = b_n$ when b_n is shifted into the structure and is zero otherwise. Q at the EXOR gates and at the adder 1 performs the 2's complement of the multiplicand if $b_n = 1$. This structure can also be easily modularized.

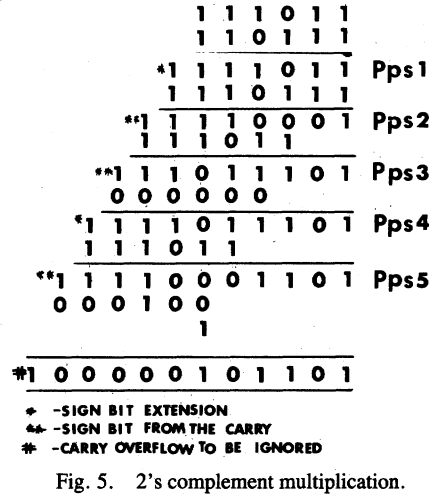

Fig. 5. 2's complement multiplication.

IV. Conclusion

This correspondence describes an FSP multiplier. When compared to conventional add–shift and related methods the multiplier presented here is about one-third faster for an about one-third increase in hardware. Also, it does not require any external correction for 2's complement multiplication. The FSP multiplier represents an attractive compromise between hardware/power and speed for a given circuit technology. Because of its reduced hardware requirement and high speed, it might prove to be one of the attractive candidates for an on-chip hardware multiplier for microprocessors and calculators. It is noted in [18] that if low-power devices such as MOSFET are used, the speed of high-speed circuits requiring large chip area such as carry lookahead adders is greatly slowed down due to parasitic capacitance caused by large fan-outs of some gates.

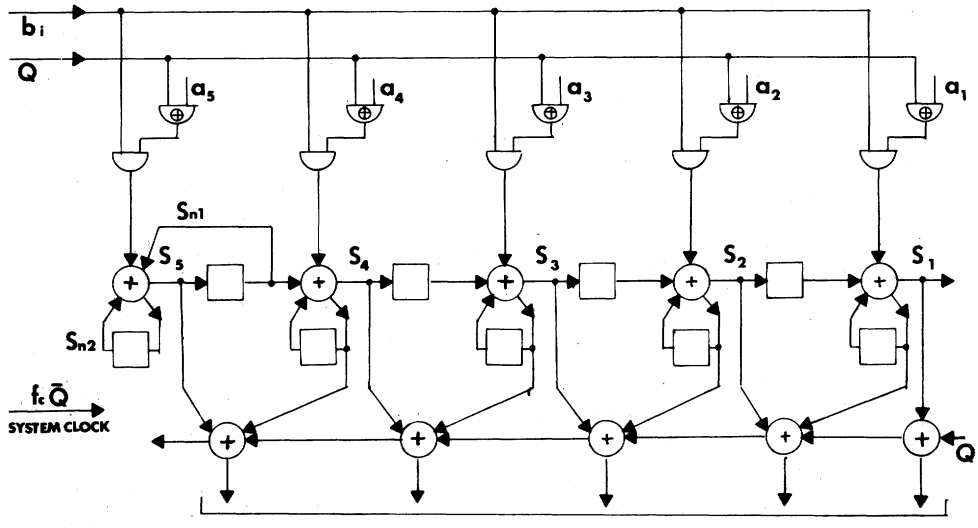

Fig. 6. 2's complement FSP multiplier.

Thus, when the chip area is limited, circuits which occupy a small area are often used for high speed. Reference [18] also cites the example that in the Intel 8080 microprocessor chip a ripple adder instead of a carry lookahead adder is used for high speed because of chip size limitation.

Compared to Booth's algorithm the 2's complement multiplication procedure presented here makes no decision in every stage about whether to add or subtract the multiplicand from the accumulated sum of partial products or to retain the previous sum. This means slightly increased speed, reduction in hardware, and fewer control lines.

The advanced Micro Devices' AM 24LS14 8 bit serial–parallel 2's complement multiplier uses Booth's algorithm internally. The multiplier constructed for n bit multiplication using AM 24LS14 requires full $2n$ clock cycles as opposed to only n clock cycles and $(n - 1)$ full adder stages of ripple-carry propagation for the method described here.

References

[1] C. S. Wallace, "A suggestion for parallel multipliers," *IEEE Trans. Electron. Comput.*, vol. EC-13, pp. 14–17, Feb. 1964.

[2] L. Dadda, "Some schemes for parallel multipliers," *Alta Frequenza*, vol. 34, pp. 349–356, Mar. 1965.

[3] A. Habibi and P. A. Wintz, "Fast multipliers," *IEEE Trans. Comput.*, vol. C-19, pp. 153–157, Feb. 1970.

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

[5] C. R. Baugh and B. A. Wooley, "A two's complement parallel array multiplication algorithm," *IEEE Trans. Comput.*, vol. C-22, pp. 1045–1047, Dec. 1973.

[6] A. Stenzel, B. Kubitz, and C. Garcia, "A compact high-speed parallel multiplication scheme," *IEEE Trans. Comput.*, vol. C-26, pp. 948–957, Oct. 1977.

[7] S. Waser, "High speed monolithic multiplier for real-time digital signal processing," *Comput.*, vol. 11, no. 10, pp. 19–29, Oct. 1978.

[8] E. L. Johnson, "A digital quarter square multiplier," *IEEE Trans. Comput.*, vol. C-29, pp. 258–261, Mar. 1980.

[9] E. L. Braun, *Digital Computer Design*. New York: McGraw-Hill, 1962.

[10] Y. Chu, *Digital Computer Design Fundamentals*. New York: McGraw-Hill, 1962.

[11] E. E. Swartzlander, Jr., "The quasi-serial multiplier," *IEEE Trans. Comput.*, vol. C-22, pp. 317–321, Apr. 1973.

[12] T. G. McDonald and R. K. Guha, "The two's complement quasi-serial multiplier," *IEEE Trans. Comput.*, vol. C-24, pp. 1233–1235, Dec. 1975.

[13] I. N. Chen and R. Willoner, "An $O(n)$ parallel multiplier with bit-sequential input and output," *IEEE Trans. Comput.*, vol. C-28, pp. 721–727, Oct. 1979.

[14] R. Gnanasekaran, "On a bit-serial input and bit-serial output multiplier," *IEEE Trans. Comput.*, vol. C-32, pp. 878–880, Sept. 1983.

[15] L. B. Jackson, S. 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.

[16] R. F. Lyon, "Two's complement pipeline multipliers," *IEEE Trans. Commun.*, vol. COM-12, pp. 418–425, Apr. 1976.

[17] C. R. Baugh and B. A. Wooley, "Digital multiplier power estimates," *IEEE Trans. Commun.*, vol. COM-23, pp. 1287–1288, Nov. 1975.

[18] A. Sakurai and S. Muroga, "Parallel binary adders with a minimum number of connections," *IEEE Trans. Comput.*, vol. C-32, pp. 969–976, Oct. 1983.

General Model for Memory Interference in Multiprocessors and Mean Value Analysis

BOHDAN ŠMILAUER

***Abstract*—This correspondence seeks to generalize and clarify the general model for memory interference (GMI) in multiprocessors as proposed by Hoogendoorn. The interference model creates a queueing network where some service centers are FCFS with constant service times; therefore, we also apply the mean value analysis (MVA) approximation suggested by Reiser to solve this model. Furthermore, we reduce the computations in this approximation by applying the iterative scheme suggested by Schweitzer. Although many authors have studied the memory interference problem, nobody has used the above-mentioned MVA approximations. We study these MVA approximations in the context of the memory interference problem and show that they produce better results with much less computation.**

Manuscript received March 12, 1984; revised January 29, 1985.

The author is with the Research Institute for Mathematical Machines, P.O. Box 65, Parlerova, 169 00 Prague 612, Czechoslovakia.

0018-9340/85/0800-0744$01.00