Power Efficient Sequential Multiplication Using Pre-computation

N. Honarmand, M.R.Javaheri, N.Sedaghati-Mokhtari and A. Afzali-Kusha
Nanoelectronics Center of Excellence, School of Electrical and Computer Engineering,
University of Tehran, Tehran, IRAN
nima@cad.ece.ut.ac.ir, {m.javaheri, n.sedaghati}@ece.ut.ac.ir, afzali@ut.ac.ir

***Abstract*—A pre-computation based technique to lower the power consumption of sequential multipliers is presented. This technique also speeds up the multiplication by reducing the number of clock ticks required to complete a multiplication. The proposed technique may be applied to different sequential multiplication schemes. The benchmark data is extracted from typical DSP applications to show the efficiency of the proposed technique in the domain of DSP computations in which the low power computing is of rapidly increasing importance. The results show an average of 25% reduction in the switching activity and 30% reduction in the clock tick count, compared to sequential multipliers without this technique.**

I. INTRODUCTION

In recent years, power consumption has become a critical design concern for many VLSI systems. Especially, it is an important bottleneck in portable battery-operated applications where the power consumption may be more important than speed and area. In CMOS technology, a great deal of power dissipation is caused by charging and discharging of the load capacitances. Therefore, it is crucial to minimize the number of signal transitions in circuits for a low power design [1].

Because of the frequent use of arithmetic units such as multipliers and adders and their high power consumption, many low-power techniques have been proposed to optimize these functional units in terms of power consumption (see, e.g., [2-5]). Among other computing systems, DSP applications make extensive use of multiply and accumulate computations. Therefore, the design and the implementation of power-efficient arithmetic units, especially multipliers, is essential for the design of low-power DSP hardware[6].

Several power reduction techniques, in different levels of abstraction (from system and architecture levels to logic and circuit levels), have been proposed in literature. Some of these approaches, such as asynchronous multiplier architecture and split registers, use on-demand computation [2]. High level optimization techniques like optimization of encoding schemes (*e.g.*, booth encoding) [3], operand representation optimization [3], structure optimization of partial product reduction circuit [3], signal gating to deactivate portions of a full-precision multiplier [3], and the use of row and column bypassing techniques in parallel array multiplier [4] have also been proposed. In the circuit level, less dissipative logics such as CPL-TG for full adder block [5] is another low power multiplication technique. In DSP applications like digital filters and FFT blocks, which involve multiplication by a fixed set of coefficients, substantial research have been devoted to topics such as coefficient optimization [6], and applying Partially Guarded Computation concept to data dominated applications [1].

From one point of view, multipliers can be categorized to sequential and combinational ones. Sequential multipliers are attractive for their low area requirements. They, however, take more time to complete a multiplication operation compared to combinational ones. In this work, we propose a pre-computation based technique to lower the power consumption of sequential multipliers. The paper is organized as follows: We describe the proposed multiplier architecture in Section II and the benchmark circuits in Section III. The results and discussion are presented in Section IV while the summary and conclusion are given in Section V.

II. PROPOSED TECHNIQUE

In a sequential multiplier, the multiplication process is divided into some sequential steps. In each step some partial products will be generated, added to an accumulated partial sum and the partial sum will be shifted (towards left or right, depending on the scheme in use) to align the accumulated sum with the partial products of next steps [7]. Therefore, each step of a sequential multiplication consists of three different operations which are generating partial products, adding the generated partial products to the accumulated partial sum, and shifting the partial sum. Fig.1 shows partial product generation and addition in a sequential multiplier.

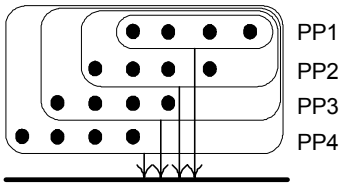

Figure 1. Row-by-row addition in a sequential multiplier

In what follows, the terms *multiplicand* and *multiplier* refer to the first and second operands of a given multiplication, respectively. In each multiplication step, one or more multiples of the *multiplicand* are generated and added to the partial sum through a two- or multi-operand addition operation.

To reduce the power consumption of the multiplier, we make use of the fact that in each sequential circuit, a great deal of power consumption is consumed by bi-stable elements, like flip flops whose dissipated power is proportional to the clock tick count. Hence, reducing the clock tick count required for the completion of a multiplication can lower the power consumption of the flip-flops and increase the speed of the circuit. This will reduce the power delay product (PDP) factor of the circuit.

The proposed technique is based on the observation that in each sequential multiplication scheme many of the generated partial products are trivial values (*i.e.*, 0) and adding them to the partial sum will have no effect on the partial sum. This phenomenon is more common in low radix multiplier (Radix-2 or Radix-4) than in high radix multipliers.

Considering these facts, one can lower the power consumption and increase the speed by eliminating all multiplication steps associated with the generation and the accumulation of trivial partial products. Consider a group of such multiplication steps at the end of the multiplication. Because all the generated partial products are 0, adding them to the accumulated partial sum will have no effect on the accumulated sum. One can skip over all such additions to save power and time. This is achieved by skipping over the whole corresponding multiplication step including the shift operations. At the end of the multiplication process, the results should be corrected by properly shifting them through a simple shift operation.

To show the basis of the work, we first introduce the concept of *Most Significant Position* (MSP). For an unsigned binary number, the MSP is the position of the highest 1 bit in the binary representation of the number, numbering the LSB with 1. For example, in case of $(12)_{10} = (00001100)_{2'sC}$ MSP is 4. For a negative number, the MSP is equal to the MSP of its absolute value. For example, in case of $(-12)_{10} = (11110100)_{2'sC}$ MSP is again 4.

Given the above definition, for a number with an MSP of m, the absolute value of the number will be less than 2^m. Now suppose that the MSPs of the two multiplication operands are *m1* and *m2* and, hence, their absolute values will be less than 2^{m1} and 2^{m2}, respectively. As a result, the product of the multiplication will have an absolute value less than 2^{m1+m2}. The MSP for the product will be at most m_1+m_2-1 and thus having, at most, this number of meaningful bits. All other higher bits will be predictable to be 0 or 1, based on the sign of the result. Therefore, one can avoid computing those higher bits, and let the final correction phase produce them. As we will see later, this final correction can be just a simple arithmetic or logical shift in most cases.

Based on the above discussion, in each multiplier implemented using the proposed technique, there are two additional phases, besides the normal multiplication steps. These phases are the pre-computation phase and the final correction phase.

A. Pre-computation Phase

In order to apply the technique to a sequential multiplication scheme, an initial pre-computation should be performed on the operands. This pre-computation determines the actual number of multiplication steps needed to compute the final result. If the calculated number is smaller than the maximum number of steps, it will lead to reduction in the required multiplication steps.

The pre-computation phase, which basically counts the number of leading zero/ones in the operands, may be implemented through a priority encoder circuit [8]. The nature of the pre-computation phase, though similar for different multiplication schemes, somewhat depends on the multiplication scheme in use. One should note that between the two operands in the multiplication process, to reduce the required clock count, the *multiplier* should be selected as the operand with the lower MSP.

In the multiplier implementations presented in this work, we have used priority encoder circuits, tailored to the specific multiplication scheme in use, to do the necessary pre-computation. Each pre-computation circuit will calculate the necessary clock count for each multiplication using the MSPs of the operands.

B. Final Correction Phase

The exact operation of the final correction phase depends on the effect of the omitted shift operations on the final result. In many cases, this final correction is just a simple logical or arithmetic shift (*e.g.*, in case of modified booth multiplier it is an arithmetic left shift and in case of shift-and-add multiplier it is a logical left shift).

Our experiments have shown that implementation of shift and rotate operations through commonly-used barrel shifters tend to create high switching activities. To remedy this problem for obtaining the correct result, we have gated the inputs of the shifters. This allows the inputs of the shifter to change only after the last multiplication step and finalization of the intermediate result.

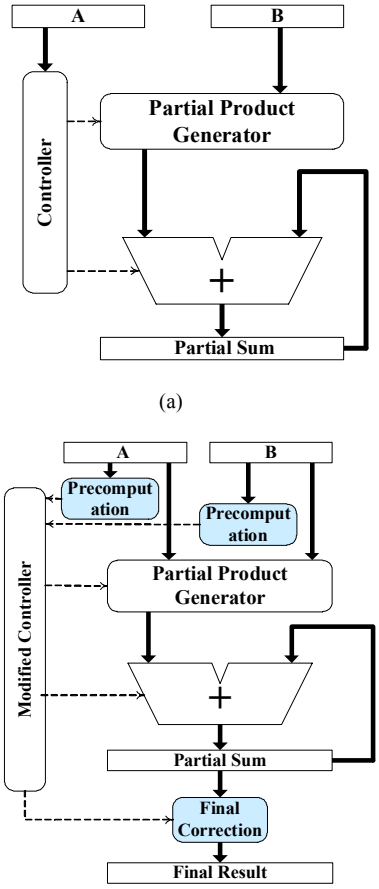

Figure 2. (a) General sequential multiplier architecture and (b) Modified sequential multiplier architecture

III. BENCHMARK CIRCUITS

We have applied the proposed technique to three 16 ×16 sequential multipliers which are signed modified booth multiplier (MBM), signed modified booth multiplier with carry save addition (CSA MBM), and unsigned shift and add multiplier (SHAM).

The general structures of the multiplier with and without the proposed technique are shown in Fig.2. Table I shows the calculated number of required multiplication steps for multipliers with different MSPs.

For the two signed multiplication schemes, the final correction phase is an arithmetic right-shift operation and for the unsigned scheme, it is a logical right-shift operation. The shift count is equal to the number of shifts that had to be performed during the omitted multiplication steps. It is the difference between the total required shifts and the number of shifts preformed during the multiplication steps.

In next section, we will describe the method used to extract benchmark data and present the experimental results.

TABLE I. REQUIRED CLOCK TICK COUNT FOR DIFFERENT MULTIPLICATION SCHEMES AND DIFFERENT MSPS

MSP	MBM	CSA MBM	SHAM
0	1	1	0
1	1	1	1
2	1	1	2
3	2	2	3
4	2	2	4
5	3	3	5
6	3	3	6
7	4	4	7
8	4	4	8
9	5	5	9
10	5	5	10
11	6	6	11
12	6	6	12
13	7	7	13
14	7	7	14
15	8	8	15
16	8	8	16

IV. BENCHMARK DATA AND EXPERIMENTAL RESULTS

To assess the efficiency of the proposed technique, we have extracted benchmark data from two typical DSP applications. We have implemented two digital filters of order 4 with the following specifications:

- An elliptic low pass filter with F_s = 11025, R_p = 1.0 db, R_s = 20.0 db, F_c = 2000 Hz.
- An elliptic band pass filter with F_s = 11025, R_p = 1.0 db, R_s = 20.0 db, F_{c1} = 2000 Hz, F_{c2} = 2500 Hz.

In the specifications above, F_s stands for the sampling frequency of the input data of the filter, R_p and R_s stand for attenuation in pass and stop bands respectively, and F_c, F_{c1}, and F_{c2} are cutoff frequencies of the two filters.

To generate the inputs of the filters, we have selected "ringin.wav" from the media files of MS Windows™ and applied the filters to this file and its scaled-up and scaled-down versions. The maximum amplitude, for the file, its scaled-up, and its scaled-down versions were 0.7, 1.0, and 0.2, respectively. We have also applied a white noise input with the amplitude of 1.0 and the power of 1.5 watts. Therefore, we will have 8 sets of data for each multiplier.

Fig.3 shows the MSP distribution of the benchmark data. The percent of switching activity and clock tick count reduction for each circuit are given in Tables II and III, respectively. In these tables, BP and LP refer to data from the band pass and low pass filters, respectively. Also, NS, US and SD refer to the data derived from non-scaled, scaled-up and scaled-down versions of "ringin.wav", respectively, and N refers to the data derived from the application of the noise input. As can be seen, depending on the MSP distribution of multiplication operands, the proposed technique leads to switching activity reductions from 16 to 39 percent and clock tick count reductions from 18 to 40 percent.

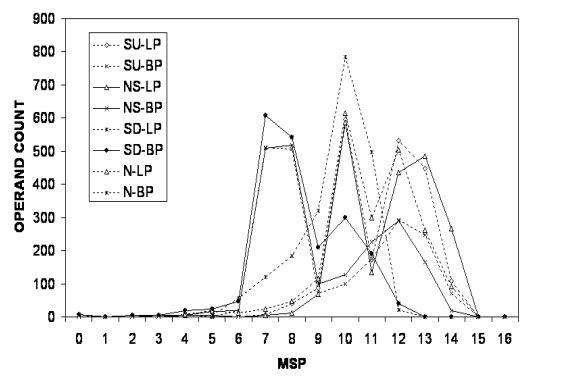

Figure 3. MSP distribution of multipliers in benchmark data

The reason we have selected the data extracted from DSP applications is that in such applications, the multiplication operands tend to have higher MSPs than common software applications which run on general purpose CPUs. Based on the data given in [2], the MSPs of the multiplication operands of typical benchmark software applications tend to be much less and, hence, the proposed technique should give rise to much higher gain in terms of power reduction and speed improvement.

V. SUMMARY AND CONCLUSION

In this paper, we proposed a pre-computation based technique for decreasing the power and increasing the multiplication speed of sequential multipliers. The proposed technique is based on the fact that all multiplication steps associated with the generation and the accumulation of trivial partial products can be eliminated at the end of the multiplication. This reduces the required clock tick count and switching activity of the multiplier.

We have applied the technique to three different multiplication schemes and presented the results obtained for these modified multipliers on some benchmark data extracted from two typical DSP applications. The results show that this technique can achieve 16 to 39 percent switching activity reduction and 18 to 40 percent clock tick count reduction, depending on the MSP distribution of multiplication operands.

REFERENCES

[1] J. Choi, J. Jeon, and K. Choi, "Power minimization of functional units by partially guarded computation," in *Proc. ISLPED*, 2000, pp. 131-136.

[2] Y. Liu and S. Furber, "The design of a low power asynchronous multiplier," in *Proc. ISLPED*, 2004, pp. 301-306.

[3] Z. Huang, "High-level optimization techniques for low-power multiplier design," PhD dissertation in Computer Science, *UCLA*, 2003.

[4] M.-C. Wen, S.-J. Wang, and Y.-N. Lin, "Low-power parallel multiplier with column bypassing", Electronics Letters, vol. 41, no. 10, pp. 581-583, 12th May 2005.

[5] I.S. Abu-Khater, A. Bellaouar, and M.I. Elmasry, "Circuit techniques for CMOS low-power high-performance multipliers," IEEE Journal on Solid-State Circuits, vol. 31, no. 10, pp. 1535-1546, Oct.1996.

[6] S. Hong, S. Kim, M.C. Papaefthymiou, and W.E. Stark, "Low power parallel multiplier design for DSP applications through coefficient optimization," in *Proc. ASIC/SOC*, 1999, pp. 286-290.

[7] B. Parhami, "Computer arithmetic: algorithms and hardware design", New York: Oxford University Press, 2000, pp. 143-145.

[8] V.P. Nelson et al, "Digital logic circuit analysis and design", Englewood Cliffs, NJ: Prentice Hall, 1995, pp. 259-264.

TABLE II. SWITCHING ACTIVITY REDUCTION IN MODIFIED CIRCUITS

Multiplier	SU-LP	SU-BP	NS-LP	NS-BP	SD-LP	SD-BP	N-LP	N-BP	Avg
MBM	16	18	18	19	28	25	16	25	21
CSA MBM	25	24	28	27	39	34	24	34	29
SHAM	18	25	20	27	29	33	18	33	25

TABLE III. CLOCK TICK COUNT REDUCTION IN MODIFIED CIRCUITS

Multiplier	SU-LP	SU-BP	NS-LP	NS-BP	SD-LP	SD-BP	N-LP	N-BP	Avg
MBM	18	28	20	29	29	35	18	35	27
CSA MBM	18	28	20	29	29	35	18	35	27
SHAM	29	38	31	40	40	40	29	45	37