Low Power Combinational Multipliers Using Data-driven Signal Gating

Nima Honarmand and Ali Afzali-Kusha
Nanoelectronics Center of Excellence, School of Electrical and Computer Engineering
University of Tehran, Tehran, Iran
nima@cad.ece.ut.ac.ir, afzali@ut.ac.ir

***Abstract*— A data driven approach to design and optimization of low power combinational multipliers is presented. This technique depends on signal gating to avoid un-necessary computations and thus reduce the switching activity and power consumption of combinational multipliers. The proposed technique can be equally well applied to signed and unsigned multiplications. At the same time, it imposes reasonable area and delay overhead on the circuit. 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 26% percent reduction in the switching activity and 22% area and 27% delay overhead, compared to combinational multipliers without this technique.**

Keywords—low power multiplication, combinational multiplier, signal gating

I. INTRODUCTION ~~(HEADING 1)~~

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 architectures 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 unnecessary 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 data-driven, signal-gating based technique for design and optimization of one class of combinational multipliers, called array multipliers [7]. The paper is organized as follows: We describe the proposed technique in Section II and the benchmark data and results in Section III, while the summary and conclusions are given in Section IV.

II. PROPOSED TECHNIQUE

Combinational multipliers are characterized with their low latency and high area requirements, compared to sequential multipliers. One of the most widely-used structures for combinational multipliers is the array multiplier which is especially attracting for its highly regular structure and lack of long wires [7]. In this class of multiplier, the multiplication is performed by consecutive addition of generated partial products. This multiplier is composed of an array of Carry-Save Adder (CSA) rows, each one adding one of the partial products to the accumulated sum of the previous partial products and one final row of Carry-Propagation Adder (CPA) to convert the accumulated sum, which is stored in a redundant number system, to the binary system. More details on this kind of multiplier can be found elsewhere [7].

As stated above, every row of CSAs will add one of the partial products to the accumulated sum. Thus if one of the partial products is zero, then the corresponding CSA row will not change the accumulated sum. If all of the partial products starting from row *r* are zero, then all the corresponding CSA rows won't do any useful computation and the final result would be like that obtained up to position *r*. Thus, it would be reasonable if someone tries to "turn-off" those CSA rows to prevent unnecessary switching activities.

One of the commonly used techniques to prevent switching activity in a combinational circuit is Signal Gating [3]. In this technique the inputs of the combinational circuit are gated so that the glitches and value changes in the previous logic levels would not propagate to next ones. There are different gating elements such as latches and AND/OR gates (ANDing with '0' and ORing with '1' will prevent signal changes to propagate to the output of the gate).

Using AND/OR gates have the drawback that, in this technique, gating may change the value of the gated signal (although just one time during every gating action) and thus may introduce unwanted switchings in the circuit. But it has the advantage of low area overhead. Using latches, on the other hand, may result in less switching activity while occupying more area. In this work we use latches for implementation purposes.

In what follows, we will use *A* (or *multiplicand*) and *B* (or *multiplier*) for the first and second operands of a given multiplication, respectively. Thus, the *i*-th partial product of a multiplication operation is zero if *i*-th most significant bit of *B* is zero; otherwise it is equal to the $A * 2^i$ $(i \geq 0)$. Also, a Gating Boundary (GB) will refer to a CSA row whose inputs are gated using latches and a Gated Segment refers to a series of CSA rows between two consecutive GBs or between the last GB and the CPA row.

One can use one or more GBs in his/her design, but some issues limit the maximum number of GBs that could be effectively used. When you decide to designate a CSA row as a GB, this implies that all the CSA rows below that point may be de-activated and thus the accumulated partial sum from that point should be transferable to the final CPA row to be converted to the binary number system. This means that you have to place a multiplexer with as many inputs as the number of GBs plus one at the input of the CPA row.

But our experiments show that large multiplexers naturally see a great deal of switching activity inside them and cause much at their outputs. Thus the number of GBs should be kept to as minimum as possible to prevent the overhead of using multiplexers from diminishing all the benefits obtained through signal gating.

The following section will demonstrate the design of a 2-GB unsigned multiplier. Unsigned multiplier will be used as a building block (with minor modifications) in the Signed, 2's complement multiplier.

A. Unsigned Multiplier Implementation

Figure 1. shows the structure of a 4×4 unsigned multiplier with 2 GBs placed at rows 1 and 3. For those full adders (FAs) placed at GBs, all the inputs of the FA are latched. The reason to latch all the inputs is that a switching on any input line may result in switching on the output lines, which in turn may propagate all the way down to the CPA row.

Assume that there are two registers at the inputs of the multiplier, holding the values of *A* and *B*. For the following discussion, suppose that the registers are sensitive to the positive edge of the clock. To achieve the best result, the GB latch control signals should be available at the positive edge of the clock, when the new operands are going to appear at the multiplier inputs. Because, otherwise, the changed inputs would introduce switchings in those segments of the circuit that should be gated before the latches could de-activate those segments. These control signals should be computed and stored in the input registers together with the input operands. In this example the latch enable signal for GB1 is *B[3]* and that of GB0 is *(B[1] OR B[2] OR B[3])*. The former indicates whether the bit at position 3 of *B* is '1' and the latter indicates that *B* has some '1' bits after position 1 (inclusive). Note that whenever GB0 is inactive, *i.e.* '0', the GB1 would also be inactive.

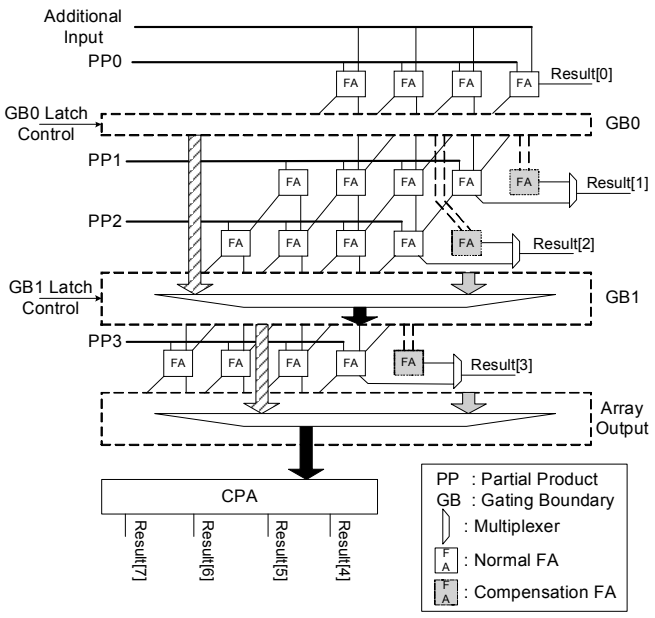

Figure 1. Structure of a 4×4 modified unsigned multiplier

To provide a regular structure and avoid long wires, at the output of each gated segment there is a 2-to-1 multiplexer that determines the input of the circuit following that segment. If the latch enable signal for that segment was active, the output of the final CSA line of the segment (gray block arrows in Figure 1.) would become the input of the next segment. Otherwise a shifted version of the segment's input (hatched block arrows in Figure 1.) will become the output of the segment.

Also each CSA row should provide one bit of the result (as is the case with ordinary array multipliers). When the containing gating segment of a CSA row is active, the data in that segment is valid and the required bit is the *sum* output of the least significant FA of the CSA row. But when the segment is inactive, the required bit should be directly computed from the inputs of the segment. Thus, there is an additional FA (Compensation FAs in Figure 1.) in every such CSA row, responsible of computing the required bit in the latter case. A 2-to-1 multiplexer chooses the *sum* output of the proper FA (Normal or Compensation FA) as the relevant bit of the final result.

As can be seen from Figure 1. , the resulted structure is still highly regular and the length of the wires are limited to the

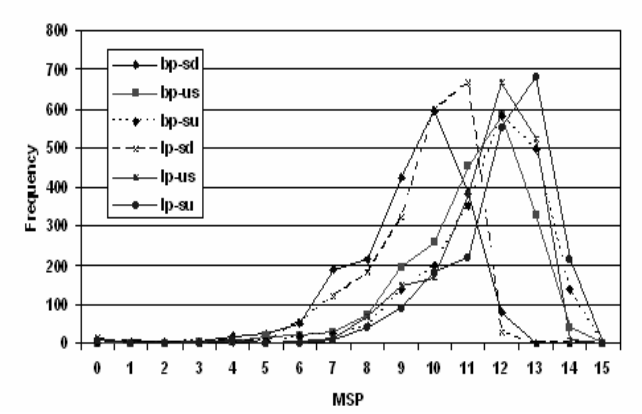

Figure 2. MSP distribution in individual benchmark datasets

distance of GBs, which tend to be small in real designs. In our experimental results it won't exceed 3 rows (See section III).

B. Signed Multiplier Implementation

The previous design for the unsigned multiplier could be easily extended, with some modifications, for signed multiplication.

First of all, the inputs to a signed multiplier are in 2's complement number system. Thus when you have a negative number with small magnitude, *e.g.* -1, there are many leading '1' bits that just represent sign extension and have no computational value. Thus we should try to avoid any computation as a result of such sign bits of *B*. In a conventional signed array multiplier, like Baugh-Wooley's [7], these leading '1' bits will take part in the actual computation of the final result and thus we can not just use a Baugh-Wooley (or similar) multiplier and apply the signal gating to it. Hence, we use the following formula to handle the case of negative value of *B*:

$$(1)\quad a \times b = -(\overline{a} \times b + b)$$

According to (1), in case of a negative value for *B* we will use the complement of *B*, which represents a positive number, in the array multiplication circuit, add *A* to the output of array, and negate the final result. Another possible technique could be to negate *B* and then negate the final result but this has the drawback that, due to rippling nature of negation operation, it will introduce lots of switching activity in the negation circuit, which in turn will cause a great deal of switching activity inside the CSA rows.

C. Selection of Gating Boundries

To describe the gating boundary selection technique, 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.

The most important task in the optimization of a multiplier is to select the number and position of GBs. To do this, one should have some intuition into the MSP distribution of the multiplication operands.

Figure 2. shows the MSP distribution in each of the benchmark datasets. **Error! Reference source not found.** shows the MSP distribution in the whole datasets. Having such MSP distribution diagrams, one can decide on the number and position of GBs. GBs should be chosen so that they can filter out some unnecessary computation in nearly all the multiplications. In **Error! Reference source not found.**, a GB at bit 13 seems to be a good choice because most of the inputs have less than 13 significant bits and a GB at 13 can de-activate 3 CSA rows (13, 14, and 15). The next candidate for GB seems to be at 10^{th} or 11^{th} rows because they still de-activate enough CSA rows and a considerable portion of the inputs have less than 10 or 11 significant bits. As the results show, the choice of (11,13) for GB positions seems to be slightly better than (10,13).

III. 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, scaled-up, and scaled-down versions are 0.7, 1.0, and 0.2, respectively.

Figure 2. and **Error! Reference source not found.** show the *multiplier* MSP distribution of the benchmark data. The percent of switching activity reduction for different GB compositions is given in TABLE I. . In these tables, BP and LP refer to data from the band pass and low pass filters, respectively. Also, NS, SU and SD refer to the data derived from non-scaled, scaled-up and scaled-down versions of "ringin.wav", respectively. As can be seen, depending on the selection of GBs and MSP distribution of *multiplier*, the proposed technique leads to switching activity reductions from 14 to 42. As can be seen, second design (which in our opinion gives the best results) reaches an average of 26 percent of switching activity reduction.

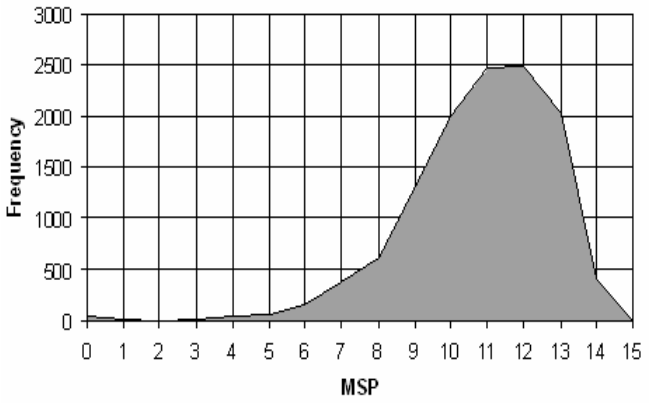

Figure 3. Total MSP distribution in benchmark datasets

The reason we have chosen the benchmark data 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 processors. 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.

Designs were synthesized in a 0.25 µm standard CMOS technology. 0 gives the area and delay overhead of the proposed technique. As can be seen, delay and area overhead of the designs are in acceptable ranges

IV. SUMMARY AND CONCLUSION

In this paper, we proposed a data driven approach for decreasing the switching activity (and hence the power consumption) of combinational array 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. The proposed method uses signal gating to de-activate those portions of the multiplier that perform no useful computation.

Also we have proposed a method to select Gating Boundaries based on the characteristics of the application data. We 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 14 to 42 percent switching activity 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.

TABLE I. SWITCHING ACTIVITY REDUCTION IN MODIFIED CIRCUITS (PERCENT)

Multiplier	SU-LP	SU-BP	NS-LP	NS-BP	SD-LP	SD-BP	Average
1 GB at 13	14	21	17	25	28	16	25
2 GBs at (11,13)	15	23	20	30	30	41	26
2 GBs at (10,13)	13	21	17	27	27	42	21

TABLE II. AREA AND DELAY OVERHAD OF MODIFIED CIRCUITS (PERCENT)

Multiplier	Area	Delay
1 GB at 13	14	23
2 GBs at (11,13)	22	27
2 GBs at (10,13)	22	27