High Performance Low-Power Signed Multiplier

Amir R. Attarha	Mehrdad Nourani
VLSI Circuits & Systems Laboratory
Department of Electrical and Computer Engineering
University of Tehran, IRAN
Email: attarha@khorshid.ece.ut.ac.ir
Email: nourani@alpha.ces.cwru.edu

M. Zakeri
Mobile Communication Industry
Iran Communication Industries (SAMA)

Abstract

In this paper, we present a high-speed low power signed multiplier with improved booth encoders and partial product generators. Our partial product generator includes only two 2-1 multiplexers in it's critical path, while previously-designed partial product generators are using three multiplexers [1] or equivalently more logic level gates [2] in their critical paths. 4:2 Compressors connected in a Wallace tree are used for adding partial products. To reduce area and improve the speed, a distributed adder is used. After the multiplier structure designed, the best logic style for this application was selected based on comparisons made by HSPICE simulations. Then, transistor sizes were optimized to obtain a high speed, low power, and area efficient multiplier in a 0.5μm CMOS process.

Keywords: VLSI Circuits, Multiplier, Booth Algorithm, Wallace Tree, Adder, and DSP Processor

0 Introduction

Since arithmetic operations dominate execution time of most DSP algorithms, a high-speed multiplier is very desirable. Currently, the multiplication time is still the dominant factor in the determination of the instruction cycle time of a DSP chip [3].

On the other hand, reducing the power dissipation in doing arithmetic operations while preserving the desired performance is indispensable for digital signal processors (DSPs), reduced instruction set computers (RISCs), microprocessors, and similar systems.

This paper describes a 17×17 bit signed parallel multiplier which works in 3.9 ns with a 0.5μm CMOS process. To reduce the number of the partial products, modified booth algorithm [4] is used and for adding partial products efficiently, 4:2 compressors [5] connected as a Wallace tree [6] are considered. Further, we implement a 32-bit distributed adder for generating the final result. After deciding on the multiplier architecture, we have compared different logic styles for multiplier implementation and have concluded that Complementary Pass-transistor Logic style (CPL [7], [8]) leads to the most efficient multiplier in terms of power and delay. For this purpose, we have employed a 0.5 μm CMOS technology with a 3.3V supply voltage.

The multiplier architecture is described in Section 2. Section 3 explains the circuit design of the basic components. Details of layout design and selection of optimum logic style are described in Section 4. Section 5 concludes the paper.

1 Architecture

One of the solutions for realization of high-speed multipliers is to enhance parallelism and to decrease the number of subsequent calculation stages. It is well known that both modified Booth algorithm and the Wallace tree are effective in decreasing calculation stages. The Booth algorithm has been widely used in parallel multipliers. In the Wallace tree array, it is well known that it is the most effective method in reducing propagation stages in addition of partial products.

The block diagram of the designed 17×17-b multiplier is shown in Fig. 1. It employs the modified Booth algorithm, a Wallace tree, and a distributed adder.

2-1 Sign-Select Booth Algorithm

The use of Booth algorithm reduces the numbers of partial products by half. Consider the multiplication of two n-bit numbers A and B in 2's complement. The multiplicand and multiplier bits are represented by a_i and b_j variables, respectively. The truth table of the conventional modified Booth algorithm is summarized in Table 1, and the related Booth Encoder circuit is shown in Fig 2. In the conventional modified Booth algorithm, three signals, X_j, $2X_j$, and M_j, are generated from three adjacent multiplier bits, b_{j-1}, b_j, and b_{j+1} for selecting a partial product, that is, one of 0, +A, -A, +2A, or -2A. Here A is the multiplicand value of n bit width. The X_j and $2X_j$ signals show whether or not the partial product is doubled and an active M_j means that the negative partial product should be used.

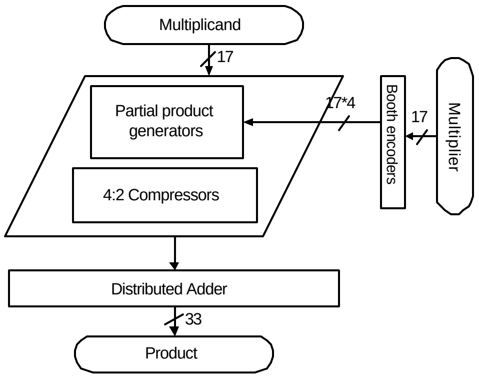

Fig. 1. Block diagram of the designed multiplier.

In this algorithm, a logical equation for the output signal, $P_{i,j}$, of the partial product generator at the *ith* multiplicand bit, a_i, and the *jth* multiplier bit, b_j, is given by

$$P_{i,j} = \left(a_i \cdot X_j + a_{i-1} \cdot 2X_j\right) \oplus M_j \quad \left(i = 0,1,2 \ldots, n-1 \qquad j = 0,2,4,\ldots,n-2\right). \tag{1}$$

which means a_i or a_{i-1} are selected depending on X_j or $2X_j$ being active, and if M_j=1, its 1's complement is computed as well. A circuit implementation of the above equation is shown in Fig 3.

Table 1. Truth table of the conventional modified Booth algorithm.

The Recoded Operand	**Multiplier Bit Triple**
0	000
1A	001
1A	010
2A	011
-2A	100
-A	101
-A	110
0	111

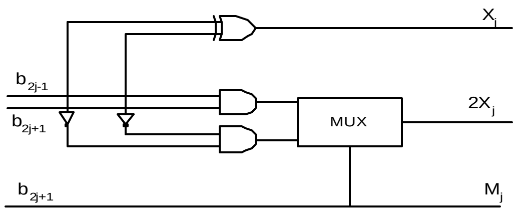

Fig 2. Conventional Booth's Encoder circuit.

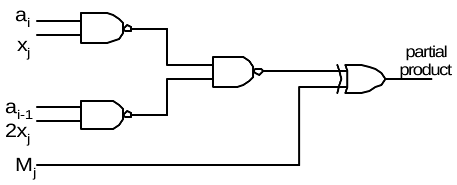

Fig 3. Conventional partial-product generator circuit.

We have used a new Booth encoding algorithm [1] with four output signals for each stage. We have also added an extra input control signal “Sgn” to directly implement either A×B or -A×B within the architecture. An extra output signal PL_j is also provided to represent the selection of the positive partial product (Table 2.). We introduce a complex Booth encoder as shown in Fig.4. This new Booth encoder has a much simpler VLSI implementation in comparison to previous work. In the conventional encoder, only M_j, which represents the negative partial product, is generated[1]. Conversely, in our Booth encoder, signals for both negative and positive partial products are generated. PL_j and M_j become active when the partial product is positive or negative respectively. X_j and $2X_j$ signals show whether or not the partial product is doubled. Note X_j is complement of $2X_j$ in this new Booth encoding algorithm.

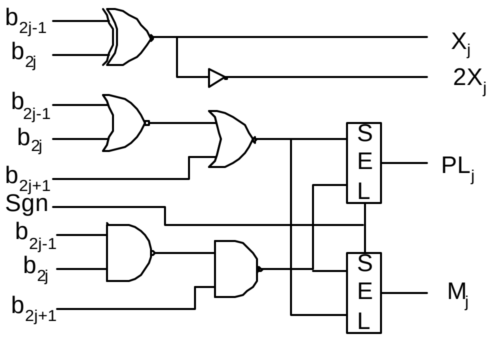

Fig4. Sign select Booth encoder circuit.

Using this booth encoder, not only we can compute either A×B or -A×B, but also we have simplified the partial product generator circuit. As shown in Fig 5. new partial product generator is much simpler than the conventional one (Fig 3.). The partial product generator is designed using three 2-1 multiplexers. In this circuit the critical path includes only two 2-1 multiplexer, while previously designed partial product generators are using three 2-1 multiplexers [1] or XOR gate and actually 4 primitive gates [2] in their critical paths. Because of the large number of partial product generators used in the multiplier circuit, simplicity of this circuit leads to a lower area and power consumption.

[1]- Note that PL_j is not invert of M_j when all three input bits are equal.

Table 2. Truth table of sign select Booth encoder.

Inputs					Sign Select			
Sgn	b_{j+1}	b_j	b_{j-1}	Func	X_j	$2X_j$	PL_j	M_j
0	0	0	0	0	0	1	0	0
0	0	0	1	+A	1	0	1	0
0	0	1	0	+A	1	0	1	0
0	0	1	1	+2A	0	1	1	0
0	1	0	0	-2A	0	1	0	1
0	1	0	1	-A	1	0	0	1
0	1	1	0	-A	1	0	0	1
0	1	1	1	0	0	1	0	0
1	0	0	0	0	0	1	0	0
1	0	0	1	-A	1	0	0	1
1	0	1	0	-A	1	0	0	1
1	0	1	1	-2A	0	1	0	1
1	1	0	0	+2A	0	1	1	0
1	1	0	1	+A	1	0	1	0
1	1	1	0	+A	1	0	1	0
1	1	1	1	0	0	1	0	0

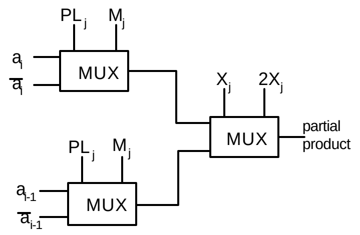

Fig 5. The new partial product generator circuit introduced in this paper.

2-2 Wallace tree Configuration

We have used a Wallace tree architecture for adding partial products efficiently. K input Wallace tree is a bit-slice summing circuit, which produces the sum of k bit-slice inputs [6]. In Wallace tree configuration, for summing the partial products produced during the multiplication, a 4:2 compressor [5] is used instead of a full adder. This will increase the efficiency and performance. As shown in Fig.6 4:2compressor has five inputs (I0, I1, I2, I3, Cin) and three outputs (Sum, Co1, Co2). It can compress four partial products into two new concurrent partial products. By using this 4:2 compressor, only three subsequent stages are needed to sum partial products of this multiplier. Each 4:2 compressor is composed of two serial full adders as shown in Fig.6. Notice that the two outputs of the 4:2 compressor (Co1, Co2) have the same weights and Co2 does not depend on Ci.

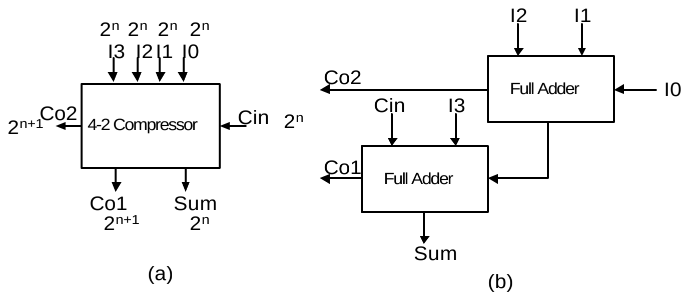

Fig.6. 4:2 Compressor. (a) Schematic. (b) Full-adder based configuration

The 4:2 compressor circuit is designed by using 2-1 multiplexers (Fig. 7). In this configuration, the critical path includes only three 2-1 multiplexers.

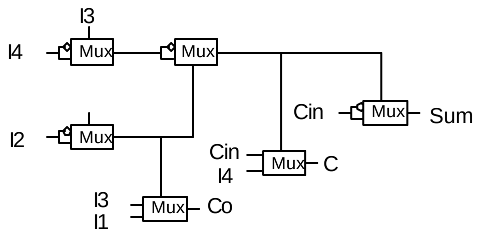

Fig 7. 4:2 compressor circuit using pass-transistor multiplexers.

For increasing parallelism and reduced delay we have used sign extension technique instead of sign propagate or sign generate methods [9].

2 Adder Structure

For generating the final product, we have designed a fast binary adder [10]. Outputs of Wallace tree are not generated simultaneously. Some outputs pass through a smaller path than the critical path. Therefore, some of the early inputs of the adder are ready after passing through at most one 4:2 compressor. However, the final inputs of the adder are ready after passing through three levels of 4:2 compressors. Considering this point, we can partition the 32-bit required adder into an 8-bit ripple carry adder and a 24-bit fast adder. 8-bit simple adder calculates its outputs while the three levels of 4:2 compressors are doing their calculations. This partitioning, not only reduces the area, but also consumes a lower power, without increasing the critical path delay.

We have utilized a fast 24-bit adder [10] in the critical path. This adder is constructed by using the principal of conditional carry generation [9]. Unlike the conventional conditional carry adders that generate all carry bits, this adder will generate only the selected carry bits. Only c_2, c_6, and c_{14} have to be generated for 24-bit addition. For generating sum bits, carry select adders [9] are used.

3 Implementation Strategy

4-1 Logic Style Choice

The increasing demand for low-power and high-speed arithmetic units can be addressed at different design levels, such as architectural, circuit, and the layout levels [8]. At the circuit level, considerable potential for power and delay savings exist by means of proper choice of a logic style (such as conventional CMOS, DPL [2], [11], and CPL [7], [8]) for implementation of digital circuits. This is because almost all of the important parameters, such as power dissipation, switch capacitance, transition activity, and short-circuit currents are strongly influenced by the chosen logic style [12].

Accordingly, the circuit delay is determined by the number of transistors in series, transistor sizes (i.e. channel widths), and loading capacitances. Circuit size depends on the number of transistors and their sizes and also on the wiring complexity. Power dissipation is determined by the switching activity and the node capacitances (made up of gate, diffusion, and wire capacitances) [13]. These important factors are strongly dependent on the logic style. Therefore, choice of suitable logic style for the 17×17-b multiplier is considered as an important step, which determines efficiency of the multiplier.

Three logic styles are commonly used in implementation of high-speed arithmetic units. These are conventional CMOS, Complementary Pass transistor Logic (CPL), and Double Pass transistor Logic (DPL). Logic gates in conventional CMOS are built from an NMOS pull-down and a dual PMOS pull-up logic network. In addition, transmission gates are often used for implementing multiplexers and XOR-gates efficiently. CMOS logic style has high robustness against voltage scaling and transistor sizing. The layout technique of CMOS gates is straightforward and efficient. An often-mentioned disadvantage of conventional CMOS is the number of PMOS transistors, resulting in high input loads and large area. Another drawback of CMOS is the relatively weak output driving capability due to series transistors in the output stage [12].

CPL is a two-rail logic style. A CPL gate consists of two NMOS logic networks (one for each signal rail), two small pull-up PMOS transistors for swing restoration, and two output inverters for the complementary output branches. The advantages of the CPL style are the small input loads, the efficient XOR and multiplexer gate implementations, the good output driving capability due to the output inverters, and the fast differential stage due to the cross-coupled PMOS pull-up transistors. One drawback of the CPL logic style is its layout complexity [12].

In the DPL logic style both NMOS and PMOS networks are used in parallel. This provides full swing to the output signals and circuit robustness. However, the number of transistors, especially large PMOS transistors, and the number of nodes is quite high [12]. DPL is a modified version of CPL that meets the requirements of the reduced supply voltage designs.

Detail HSPICE simulation of a complex circuit such as a 17×17-bit multiplier is slow and difficult to use in the initial stages of the design process. Fortunately, the multiplier architecture is a regular array of identical cells. It is therefor possible to replace most of the cells by their equivalent input capacitances and to study the performance of only a few of the basic building blocks under appropriate loading conditions.

Accordingly, conventional CMOS, DPL, and CPL are three suitable candidates for implementing the macro-cells of the multiplier. Therefore, the basic building blocks of the multiplier i.e. the sign select booth encoder (BEN), partial product generator (PPG), and 4:2 compressor, are simulated with different logic styles. The results of simulation that should be shown in next Section shows the CPL is the desired one.

4-2 Layout Design

The stated logic-design styles also strongly influence the final layout of the chosen circuit. So, we have first simulated the building blocks of the multiplier circuit with different efficient logic styles while considering their layouts and then the desired one is selected for our application.

We have drawn the layout of this multiplier using a 0.5-μm CMOS technology that is shown in Fig. 8. The major process parameters are summarized in Table. 3. We have used only two metal layers of the process.

Table. 3. Process technology

Technology	0.5μm CMOS
Gate Length	0.5μm
Gate Oxide	96Å
Supply Voltage	3.3

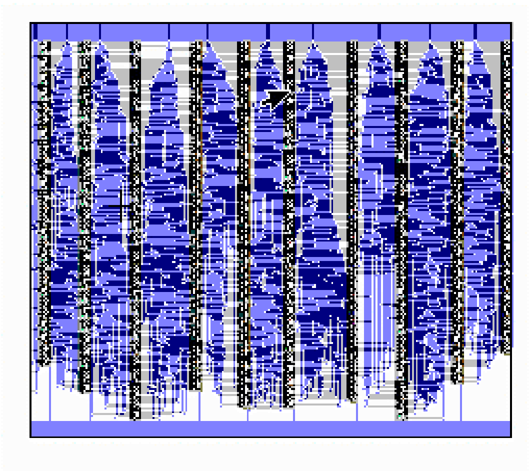

Fig. 8. Layout of the designed multiplier

4-3 Estimation of Layout Capacitance

The load observed at a layout level in simulation of a circuit consists of several parts: input capacitances of the output gates, and wiring and parasitic capacitances of the drain and source [13]. These are calculated as shown below,

$$C_L = \sum_i C_{g_i} + \sum_m C_{d_m} + \sum_k C_{wire_k} \tag{2}$$

$$C_{g_i} \approx \frac{\varepsilon_0 \varepsilon_{sio_2}}{t_{ox}} A_{g_i} \tag{3}$$

where A_{gi} is the area of the gate of transistor i,

$$C_{d_m} = C_{ja}(a_m b_m) + C_{jp}(2\,a_m + 2\,b_m) \tag{4}$$

where C_{ja} is the junction capacitance per unit area, C_{jp} is the periphery capacitance per unit length, a_m is the width and b_m is the length of the diffusion region of the transistor m.

4 Experimental Results

To obtain the best possible building blocks, we have modeled all of the alternatives with HSPICE by considering the associated capacitances. Then the channel width of the related transistors have been varied between 1 to 10 microns. The best results for each of the building blocks with considering different logic styles have been summarized in Table. 4. Table. 5. shows the efficiency of the new partial product generator in comparison to the conventional one [2].

Table. 4. Characteristics of the basic building blocks of the designed multiplier with different logic styles.

	CMOS						CPL						DPL					
	# of transistors		Channel width		Delay (ps)	Power µW	# of transistors		Channel width		Delay (ps)	Power µW	# of transistors		Channel width		Delay (ps)	Power µW
	N	P	N	P			N	P	N	P			N	P	N	P		
BEN	25	25	2	3	874	830	34	12	2	2	759	943	29	29	2	2	745	1562
PPG	12	12	2	2	786	110	14	4	2	2	418	96	20	20	2	2	862	195
COMP	40	40	2	2	710	280	23	6	2	2	485	236	30	30	2	2	636	528

Table. 5. Simulation comparison between conventional and the new partial product generator.

	# of transistors		Channel width		Delay (ps)
	N	P	N	P	
Conv PPG	12	12	2	2	786
The new PPG	14	4	2	2	418

Advantages of using distributed adder are demonstrated in Table. 6.

Table. 6. Distributed CMOS adder in comparison to a conventional adder.

	# of transistors		Channel width		Delay (ns)
	N	P	N	P	
Conventional dder (CMOS)	3646	3646	1	2	2.1
Distributed adder (CMOS)	2569	2569	2	2	1.5
Distributed adder (CPL)	2298	1028	2	2	1.1

Final specifications of the designed multiplier in different logic styles are shown in Table. 7.

Table. 7. Specifications of the designed multiplier with different logic styles.

CMOS		CPL		DPL	
# of transistors	Delay (ns)	# of transistors	Delay (ns)	# of transistors	Delay (ns)
17292	6.8	11292	3.9	20276	6.5

5 Conclusion

We have developed a new partial product generator that reduces the total area and power of our multiplier, without increasing the delay. Further, We have utilized a novel distributed adder, which includes an 8-bit simple and small ripple carry adder and a 24-bit fast adder. Using this adder, has reduced power and delay considerably. After designing multiplier at an architectural level, we have selected the CPL, the best logic style for this application, for implementation of the multiplier. During selection of the logic style, size optimization on the transistors has also been performed.

Detail HSPICE simulation has shown that CPL logic style can be considered as a high-speed, low-power style for implementation of arithmetic units in this work. Experimental results on implementation of the multiplier using a 0.5µm CMOS process concludes this paper.

References

[1] Gensuke Goto, et. al., "A 4.1 ns compact 54×54-b multiplier utilizing sign-select booth encoders," IEEE J. Solid-State circuits. vol. 32, No. 11, pp. 1676-1681, Nov. 1997.

[2] Norio. Ohkubo, et. al., "A 4.4 ns CMOS 54×54-b multiplier using pass-transistor multiplexer," IEEE J. Solid-State circuits. vol. 30, No. 3, pp. 251-256, March 1995.

[3] S. Y. Kung, "VLSI array processors," Printice Hall, 1988.

[4] A. D. Booth, "A signed binary multiplication technique," Quart. J. Mech. Appl. Math., vol. 4, pp. 236-240, 1951.

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

[6] Masato. Nagamatsu,et. al., "A 15-ns 32×32-b CMOS multiplier with an improved parallel structure," IEEE J. Solid-State circuits. vol. 25, No. 2, pp. 494-497, April. 1990.

[7] K. Yano, et. al., "A 3.8 ns CMOS 16×16-b multiplier using complementary pass-transistor logic," IEEE J. Solid-State circuits. vol. 25, No. 2, pp. 388-393, April. 1990.

[8] A. P. Chandarkasam and R. W. Brodersen, "Low power digital CMOS design," Norwell, MA:Kluwer, 1995.

[9] K. Hwang, "Computer arithmetic: principles, architecture, and design," John Wiley and Sons, 1979.

[10] Jien-Chung Lo, "A fast binary adder with conditional carry generation," IEEE Trans. on Computers, Vol. 46, No. 2, pp. 248-253, Feb. 1997.

[11] M. Suzuki, et. al., "A 1.5 ns 32b CMOS ALU in double pass-transistor logic," in Proc. 1993 IEEE Int. Solid-Statecircuits Conf., Feb. 1993, pp. 90-91.

[12] R. Zimmermann and W. Fichtner, "Low-power logic styles: CMOS versus pass-transistor logic," IEEE J. Solid-State circuits. vol. 32, No. 7, pp. 1079-1090, July. 1997.

[13] M. S. Elrabaa, I. S. Abu-Khater, and M. I. Elmasry, "Advanced low-power digital circuit techniques," Kluwer Academic publishers, 1997.