Circuit and Architecture Trade-offs for High-Speed Multiplication

Paul J. Song and Giovanni De Micheli, *Senior Member, IEEE*

***Abstract*—We consider VLSI implementations of high-performance parallel multipliers. We analyze circuit building blocks required for partial-product reduction and we propose two new schemes leading to highly regular layouts. We consider the circuit implementations related to the first scheme in three different BiCMOS technologies. We compare the die size and performance for nominal design rule values and we study the trend in scaling the feature sizes. We consider then the second scheme and we report on a silicon implementation of a prototype slice of an IEEE double-precision floating point multiplier in a 0.8-μm double-metal BiCMOS technology.**

I. Introduction

FAST arithmetic circuits are key elements of high-performance computers and data-processing systems. We are investigating the performance limitations of arithmetic units designed with different circuit technologies. In particular, we have chosen parallel multipliers as a vehicle for understanding the requirements of a CMOS or BiCMOS technology to achieve a desired level of performance.

Parallel multipliers are interesting in several respects. First, they are large circuits, where the effect of circuit design as well as the properties of the interconnect affect the circuit performance. Second, they can be implemented by regular structures. Therefore, automatic synthesis techniques can be used. In addition, the interconnect delay can be related directly to the circuit size and shape. Third, a major portion of a multiplier, the partial-product reduction tree, has a circuit structure where each cell has unit fan-out. Therefore, the cell output capacitances are mostly dominated by the length of the interconnection wires, whose length can be predicted due to the circuit regularity.

For these reasons, we have decided to investigate the partial-product reduction circuits of the multiplier. We consider floating-point double-precision multiplication, in conformance to ANSI/IEEE Standard 754 [1]. Such an operation requires an integer multiplication of two 53-b significands. This operation determines the overall speed of the multiplication.

Manuscript received February 22, 1991; revised May 17, 1991. This work was supported by the National Science Foundation under a Presidential Young Investigator award and under Grant MIP 88-22961.

The authors are with the Center for Integrated Systems, Stanford University, Stanford, CA 94305.

IEEE Log Number 9101730.

Several schemes have been used for the multiplication of the mantissa. In some cases, iterative schemes have been used [2]. Fully parallel techniques, as proposed by Wallace [3] and Dadda [4], were implemented only for shorter mantissa values. Recently, IBM RISC 6000, which has a full 56-b combinatorial multiplier, was built mostly with (7,3) counters to simplify the tree wiring [5]. Intel's 80860 chip uses (4,2) counters to reduce a 27·53-b multiplier tree [6].

In this paper, we consider the design of the partial-product reduction circuit as a way of understanding the performance limitation of multipliers. We review first the circuit structures and the basic cell circuit designs. Second, we present two novel architectures. We describe then how we used synthesis techniques to lay out such circuits in different CMOS/BiCMOS technologies. We analyze the performance of the synthesized layouts and point out their limitations. Finally, we comment on a silicon implementation of one of these structures and on the experimental results achieved in testing the circuit.

II. Architectures for High-Speed Multiplication

We consider fully parallel partial-product reduction in two stages: a combinational tree circuit followed by a fast adder. The first stage reduces the partial products to two rows which are summed by the second stage to yield the result. We have focused our attention on the first stage, because adders and rounding algorithms have been widely investigated [7]–[10].

Integer arithmetic multiplication of two unsigned n-bit numbers can be envisioned as a summation of partial products. The partial-product terms are generated by simply ANDing multiplicand bits and multiplier bits. For example, the 8-b multiplication is shown in Fig. 1(a) where the tree height (i.e., the number of partial-product rows) is eight and the total number of partial-product bits is $8 \times 8 = 64$.

It is possible to use some encoding techniques that reduce the total number of partial products. One commonly used scheme, known as the modified Booth's algorithm, can reduce the partial products nearly by half. For example, the 8-b multiplication with 2-b modified Booth's algorithm is shown in Fig. 1(b), where the tree height is five and the total number of partial-product bits is 56.

```
                           multiplicand x7   x6   x5   x4   x3   x2   x1   x0
                           multiplier   y7   y6   y5   y4   y3   y2   y1   y0
--------------------------------------------------------------------------------
                                        z07  z06  z05  z04  z03  z02  z01  z00
                                   z17  z16  z15  z14  z13  z12  z11  z10
                              z27  z26  z25  z24  z23  z22  z21  z20
                         z37  z36  z35  z34  z33  z32  z31  z30
                    z47  z46  z45  z44  z43  z42  z41  z40
               z57  z56  z55  z54  z53  z52  z51  z50
          z67  z66  z65  z64  z63  z62  z61  z60
     z77  z76  z75  z74  z73  z72  z71  z70
--------------------------------------------------------------------------------
w15  w14  w13  w12  w11  w10  w9   w8   w7   w6   w5   w4   w3   w2   w1   w0
```

where $z_{ij} = y_i$ AND x_j

(a)

```
                           multiplicand x7   x6   x5   x4   x3   x2   x1   x0
                           multiplier   y7   y6   y5   y4   y3   y2   y1   y0
--------------------------------------------------------------------------------
                    n0   s0   s0   z08  z07  z06  z05  z04  z03  z02  z01  z00
               1    n1   z18  z17  z16  z15  z14  z13  z12  z11  z10       s0
     1    n2   z28  z27  z26  z25  z24  z23  z22  z21  z20       s1
n3   z38  z37  z36  z35  z34  z33  z32  z31  z30       s2
z47  z46  z45  z44  z43  z42  z41  z40       s3
--------------------------------------------------------------------------------
w15  w14  w13  w12  w11  w10  w9   w8   w7   w6   w5   w4   w3   w2   w1   w0
```

where z_{ij} = decoding function($\mathbf{b}_i, x_j, x_{j-1}$)

$\mathbf{b}_i$ = encoding function($y_{2i-1}, y_{2i}, y_{2i+1}$)

($\mathbf{b}_i$ is a vector)

(b)

Fig. 1. (a) Two 8-b unsigned integer multiplication. (b) Two 8-b unsigned integer multiplication with modified Booth's encoding.

In ANSI/IEEE Standard 754 for binary floating-point arithmetic, the double-precision format (64-b) has a significand of 53 b. In such a case, the number of partial-product bits is reduced from $53 \times 53 = 2809$ to 1536 and the tree height is reduced from 53 to 27 by using the modified Booth's algorithm.

After the partial products have been generated, circuits called counters are used to reduce the tree height from n to 2. The final result is then computed by using a carry-lookahead (CLA) adder on the two remaining partial-product rows. We consider now the strategies for reducing the partial products.

a) The most common approach is to use carry-save adders. This is known as the Wallace tree reduction [3]. In general, the number of carry-save adder stages required to reduce the Wallace tree height from n to 2 is

$$S = \log_{1.5}(n/2)$$

$$= [\ln(n/2)]/[\ln 1.5]$$

$$= 2.4664^*[\ln(n/2)].$$

For IEEE Standard 754 without Booth encoding, the height $n = 53$ and $S = 8.083$. Therefore, we need at least nine stages, while for IEEE Standard 754 with Booth encoding, the height $n = 27$ and $S = 6.419$. Therefore, we need at least seven stages. It is important to realize that the total delay is not only well correlated with the number of stages S but also depends strongly on the delay associated with the interconnection wiring capacitance.

b) Other reduction schemes are reported by Waser and Flynn [11]. The basic building blocks are called counters (also called compressors [5] or adders [6]). An (n,m) counter is a combinational logic circuit with n inputs and m outputs. The outputs are binary encoding of the sum of the input bits. A carry-save adder is a (3,2) counter. A useful counter for multiplication is the (7,3) counter. The notation of counters can be extended to multiple weighted input columns, as in the case of the (5,5,4) counter [11].

In some occasions, counter cells with $m > \log_2(n)$ have also been used by broadening the notation. One example is the (4,2) counter. In such cases, the circuit has additional outputs so that the sum of the input bits can be represented [2].

III. Circuit and Components for Multipliers

We assume a regular implementation of the partial-product reduction circuit as an array of vertical slices. Each slice contains counter cells and the Booth decoder cells. Both encoders are at the periphery of the array. Since there are 106 slices in a full IEEE Standard 754 multiplier implementation, the horizontal dimension of the slice should be as small as possible.

We have considered CMOS/BiCMOS technologies because they support efficient design of parallel multipliers. Indeed, the CLA adder [8] and the Booth encoder/drivers can take advantage of the properties of the bipolar transistors. In this paper we concentrate on the design of the partial-product reduction array, where the use of bipolar transistors is limited to the counter output stages.

We now consider the different circuits.

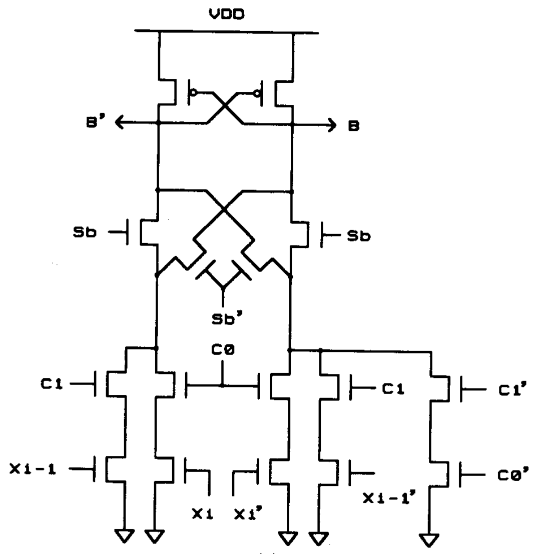
A. Modified Booth's Encoding/Decoding Scheme

For modified Booth's algorithm, the significand of the multiplier is divided into substrings of 3 b, with adjacent groups sharing a common bit. Fig. 2 shows two modified Booth's encoding schemes, where S_b denotes the sign bit, C_1 the control bit 1, C_0 the control bit 0, NZ+ the nonzero plus, NZ− the nonzero minus, and D the double. The first encoding scheme, which encodes S_b, C_1, and C_0, is a standard one [12]. However, the corresponding decoders become the critical horizontal dimension for the array slice. Therefore, a second encoding scheme, which encodes NZ+, NZ−, and D, is proposed. Its corresponding decoder can be implemented as a cascade of two circuits, where each one has less than three pairs of inputs as shown in Fig. 3(b). Fig. 3(a) shows the standard decoder. Since C_1 and C_0 are related to both x_i and x_{i-1}, there is no easy way to partition it into blocks with less than four pairs of inputs. Hence, the Booth decoder pitch on the horizontal direction is larger than Fig. 3(b).

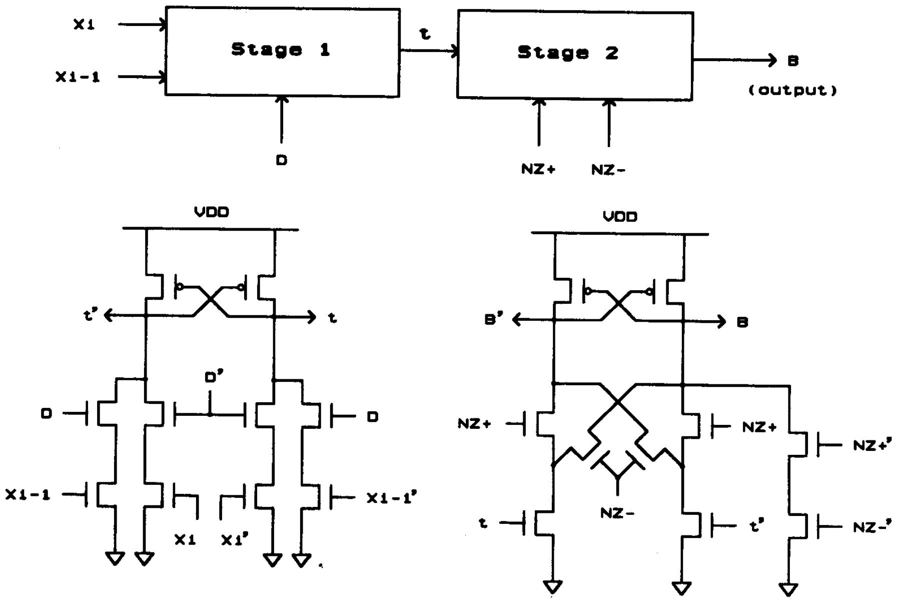
It is important to realize that Booth encoding signals and their complements have to traverse across the whole partial-product tree array. The capacitive load is high (typically 5–10 pF). Therefore, bipolar drivers should be used for the encoders when using BiCMOS technology.

Bit			Operation	Standard Encoding			Improved Encoding		
2	1	0.5							
Yi+1	Yi	Yi-1		Sb	C1	C0	NZ+	NZ-	D
0	0	0	+0	1	0	0	0	0	X (Don't care)
0	0	1	+X	1	0	1	1	0	0
0	1	0	+X	1	0	1	1	0	0
0	1	1	+2X	1	1	0	1	0	1
1	0	0	- 2X	0	1	0	0	1	1
1	0	1	- X	0	0	1	0	1	0
1	1	0	- X	0	0	1	0	1	0
1	1	1	- 0	0	0	0	0	0	X (Don't care)

Fig. 2. Standard and improved encoding using modified Booth's algorithm.



(a)



(b)

Fig. 3. (a) Standard decoding circuit for modified Booth's algorithm. (b) Improved decoding circuit for modified Booth's algorithm.

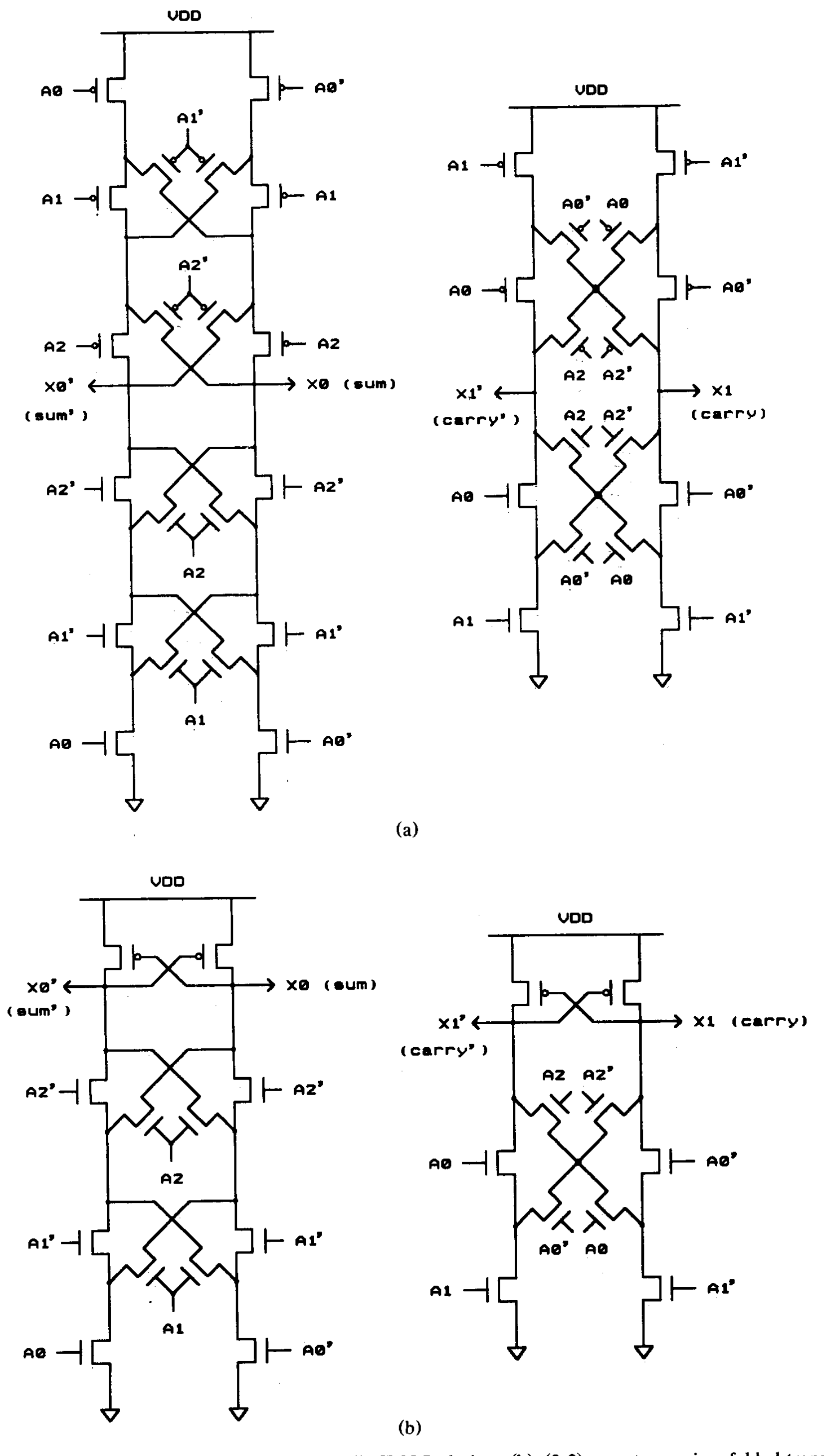


Fig. 4. (a) (3,2) counters using folded-transistor full CMOS design. (b) (3,2) counters using folded-transistor with cross-coupled PMOS load.

B. (3, 2) Counter Cell Design

Since the (3,2) counter is the key macrocell used in arithmetic multiplication, it is necessary to pay special attention to its design. We can separate the (3,2) counter cell design into two parts: the logic part and the output driver part.

1) The Logic Part: The logic part has three inputs, a_2, a_1, and a_0, and generates two output, x_1 (carry-out) and x_0 (sum):

$$\begin{array}{r} a_2 \\ a_1 \\ a_0 \\ \hline x_1\ x_0 \end{array}$$

where

$$x_1\,(\text{carry}) = a_2 a_1 + a_1 a_0 + a_0 a_2$$

$$x_0\ \ (\text{sum}) = a_2 \text{ XOR } a_1 \text{ XOR } a_0 = (a_2 \text{ XOR } (a_1 \text{ XOR } a_0)).$$

The logic part can be implemented in three ways.

a) Use XOR *and* NAND *gates:* The conventional way to construct (3,2) counter is to use EXCLUSIVE-OR (XOR) and NAND gates. Since we have complementary inputs for the XOR gate, we also need to produce complementary outputs for the next stage. Therefore, we need to generate both EXCLUSIVE-OR and its complement (EQUIVALENCE), which implies an extra inverter delay or double the hardware. Two XOR gates are cascaded to produce the sum term.

b) Use folded transistor full CMOS implementation: By observing that each EXCLUSIVE-OR gate has the property that the output toggles when any single input toggles, we can then construct the EXCLUSIVE-OR gate with folded transistors as shown on Fig. 4(a).

c) Use folded transistor cross-coupled PMOS load implementation: The previous approach can be made even more efficient in terms of transistor usage by observing that p-channel transistors can be replaced by one cross-coupled p-channel transistor pair [13]–[15] (Fig. 4(b)). In this case, we need to lower the voltage at the gate of the p-channel MOS transistor before the circuit can switch. The circuit must be designed with ratioed device sizes. Thus the delay is higher than in the second approach b).

Table I shows that approach c) is very competitive with approach b) when considering the circuit implementation of a (3,2) counter at the cell level. It is interesting to consider, though, the impact of the cell area on the delay when the cell is embedded in the partial-product reduction circuit. To be specific, we considered Signetics' SABRE BiCMOS technology [16], [17], which has 0.8-μm feature design rules, on a full Wallace tree reduction for that comparison. A counter is designed with approaches b) and c). A vertical slice involving seven stages of a (3,2) counter would be 20% larger for approach b). By considering the added wiring delay of the longer structures, which is 0.3 ns, the overall difference for the whole partial-product reduction between two approaches is 0.1 ns in favor of approach c).

TABLE I
COMPARISON OF (3,2) COUNTER CELL DESIGNS

	use XOR & NAND gate (Note 1)	use folded full CMOS	use folded cross-coupled p-channel
# of p-channel transistors	26	18	4
# of n-channel transistors	26	18	18
total # of transistors	52	36	22
input capacitance	highest	high	low
cell area	large	large	compact
built-in complementary	no	yes	yes
ratioless	yes	yes	no
speed (cell level)	slow	faster	fast
single-cell delay (Note 2)	0.45 ns	0.34 ns	0.37 ns
overall delay (Note 3,4)	4.4 ns slow	3.3 ns fast	3.2 ns faster

Note 1: Assume we use both XOR and EQUIVALENCE gates for EXCLUSIVE-OR logic.

Note 2: The simulation is done with Signetics SABRE BiCMOS process under the assumption that one gate's output drives one gate's input with the same size and without any extra capacitance load. With $V_{DD} = 4.5$ V, the input is assumed to be switching between 0.5 and 4.0 V in 0.05 ns. The output level is measured at 3.5 and 1.0 V, respectively.

Note 3: The simulation for overall simulation is done for a Booth decoded array with reduced tree height = 27. Seven stages of (3,2) counter levels are used. For short interconnection no buffers are used, while CMOS buffers are used for long interconnection.

Note 4: Delay includes whole tree partial-product reduction only (seven stages of (3,2) counters).

2) The Output Driver Part: The output driver part buffers the two signals (x_1 for carry-out and x_0 for sum) generated in the logic part to drive the next stage. Depending upon the interconnection wiring capacitance, we can choose one of the following three ways to drive the next stage:

a) Directly drive the next stage without any buffer. For two counter stages sitting next to each other, buffering may be superfluous.
b) Use a CMOS buffer to drive the next stage for medium-length interconnection (or for the capacitance value between 0.1 and 0.5 pF). This is shown in Fig. 5(a).
c) Use a BiCMOS buffer to drive the next stage for long interconnection wiring (or for capacitive load more than 0.5 pF). Since our logic stage has a built-in complementary output pair, it is simple to implement BiCMOS drivers, as shown in Fig. 5(b) where we use bipolar transistor to pull up the output rather than the weak p-channel transistors. One interesting feature is that our logic stage has differential input pair and does not require full V_{DD} swing.

C. Constructing High-Input Counters with (3, 2) Counters

By using Wallace's approach with the modified Booth's algorithm to implement a multiplier with the IEEE 754 floating-point standard, seven stages are required. How-

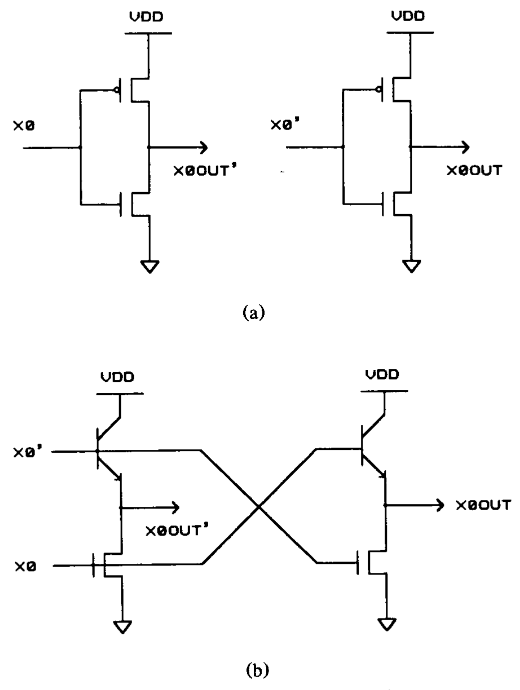


Fig. 5. (a) CMOS buffer. (b) BiCMOS buffer.

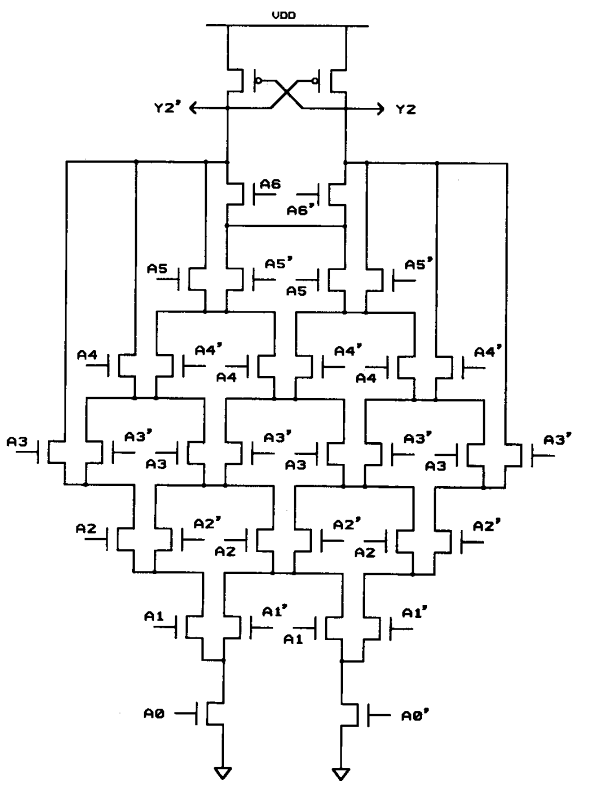


Fig. 6. Direct implementation of (7,3) counters.

```
 x0              x3            x0
 x1              x4            x1
 x2        (3,2) x5      (3,2) x2
 x3        ---------     ---------   stage 1
 x4           t3 t2         t1 t0
 x5
 x6               t0
--------          t2
y2 y1 y0   (3,2) x6
           ---------   stage 2
              t4 y0

                  t1
                  t3
           (3,2) t4
           ---------   stage 3
              y2 y1
```

Fig. 7. Constructing (7,3) counter from (3,2) counters.

```
   x5 x0             x5             x0
   x6 x1             x6             x1
   x7 x2       (3,2) x7       (3,2) x2
   x8 x3       ---------      ---------   stage 1
   x9 x4          t3 t2          t1 t0
-----------
y3 y2 y1 y0          t2             t0
                     x8             x3
               (3,2) x9       (3,2) x4
               ---------      ---------   stage 2
                  t6 t5          t4 y0

                     t1
                     t4
               (3,2) t5
               ---------   stage 3
                  t7 y1

                     t3
                     t6
               (3,2) t7
               ---------   stage 4
                  y3 y2
```

Fig. 8. Constructing (5,5,4) counter from (3,2) counters.

ever, routing may be complicated. That is the reason why high-order counters, like (7,3) and (5,5,4) counters, are used. High-input counters can be designed directly by using folded transistors, as shown in Fig. 6, where we have drawn the circuitry for output pairs y_2 and y'_2 of the (7,3) counter. This concept can be extended to the other output pairs in a straightforward way.

Such an approach has the following three disadvantages:

a) increased input capacitance—compared to the (3,2) counter, the maximum number of input transistors increases quadratically as the number of inputs increases;
b) more intermediate node capacitance—junction capacitances and transistor capacitances increase linearly as the number of inputs increases;
c) longer pull-down path—the longer the pull-down transistor path, the smaller the pull-down current is.

Due to the combination of the above three effects, the circuit slows down quadratically as the number of inputs increases.

For these reasons, it is convenient to construct high-input counters by using (3,2) counters. Throughout the rest of Section III, we shall evaluate reduction schemes with different counters by means of the number of equivalent (3,2) stage delays. The goal of this section is to describe structures that require few stage delays and lead

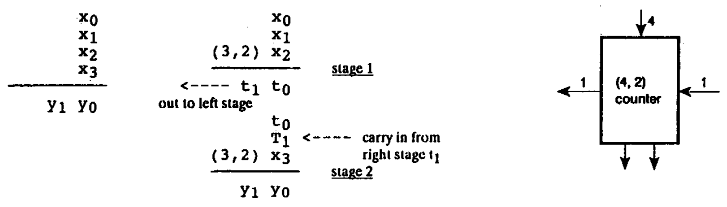


Fig. 9. Constructing (4,2) counter from (3,2) counters.

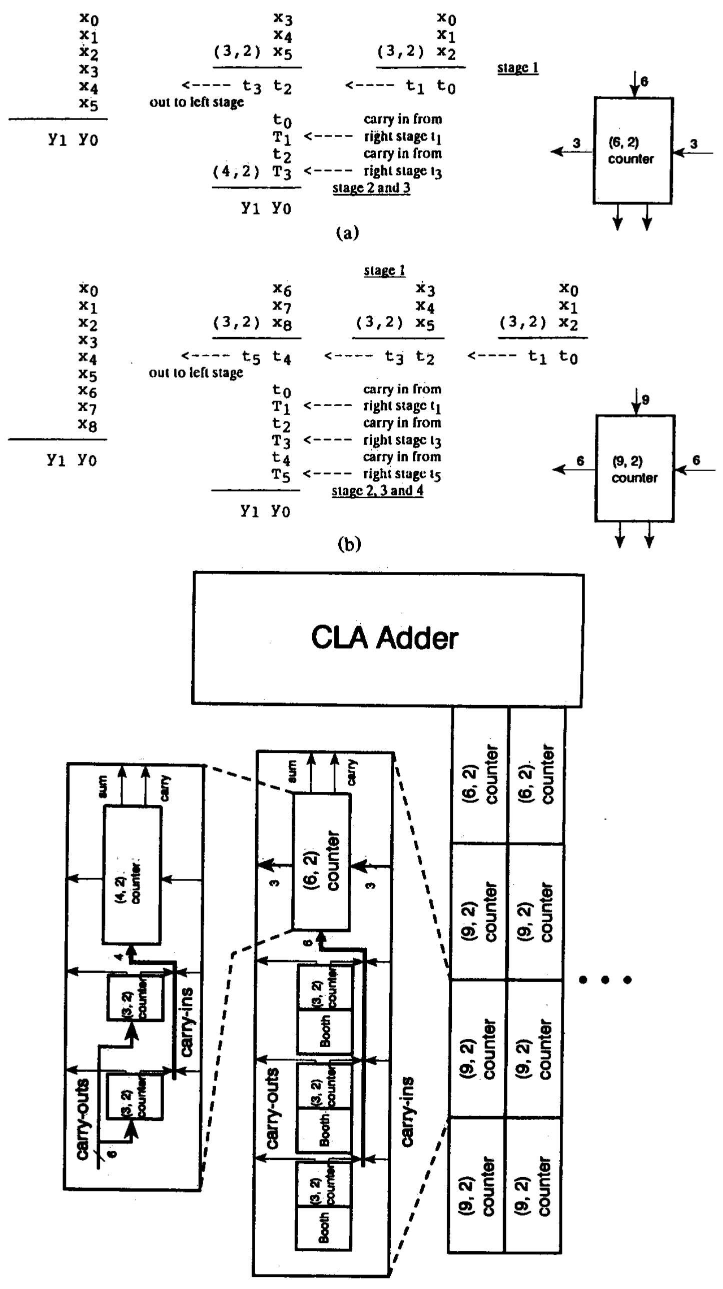


Fig. 10. (a) Constructing (6,2) counter from (4,2) and (3,2) counters. (b) Constructing (9,2) counter from (6,2) and (3,2) counters. (c) Two slices of the partial-product reduction array with the (9,2) counter family.

to highly regular layouts. A more accurate delay evaluation will be considered in Section IV in connection with layout considerations.

We consider now different counter implementations.

1) Constructing (7,3) Counters with (3,2) Counters: As shown in Fig. 7, we can group two (3,2) counters first, which generate four intermediate outputs t_3, t_2, t_1, and t_0. We then use another (3,2) counter taking x_6, t_2, and t_0 as inputs, which generates output y_0 and intermediate node t_4. Finally, we use another (3,2) counter taking t_4, t_3, and t_1 as inputs to generate the output y_2 and y_1.

Therefore, each (7,3) counter has three equivalent (3,2) counter delay stages. A partial-product reduction scheme can use (7,3) counters followed by (3,2) counters. By using modified Booth's algorithm (tree height is 27), after applying (7,3) and (3,2) counters at the first two stages, we reduce the tree height to 8. Then we can apply (3,2) counters to reduce the tree height to 6, 4, 3, and 2 in additional stages. Since the (7,3) counter is equivalent to three (3,2) counter stages, the total number of equivalent (3,2) counter stages needed is eight, one higher than the Wallace tree reduction approach. This approach yields a more regular structure than using (3,2) counters only. As a result the interconnection wiring is reduced.

2) Constructing (5,5,4) Counters with (3,2) Counters: Fig. 8 shows how we can construct the (5,5,4) counter with (3,2) counters. Again, each (5,5,4) counter has four equivalent (3,2) counter delay stages.

3) Constructing (4,2) Counters with (3,2) Counters: The (4,2) counter has a different structure [2], [18], [19]. We apply one (3,2) counter to three inputs (namely, x_2, x_1, and x_0) to yield two contemporary outputs, the temporary carry-out (t_1) and temporary sum (t_0). We pass the temporary carry-out t_1 to the *left* stage while we take the temporary carry-in T_1 from the *right* stage. Finally, we take the remaining input x_3, together with t_0, and T_1 to generate the final outputs y_1 and y_0 as shown in Fig 9.

We count two equivalent (3,2) counter delay stages for each (4,2) counter. We can use the (4,2) counters to perform the tree partial-product reduction in the following way [2]. If we start with height 27, by applying (4,2) counters repeatedly, we reduce the tree height to 14, 7, 4, and 2. Since each (4,2) counter is equivalent to two (3,2) counter stages, the total number of equivalent (3,2) counter stages needed is also eight, one higher than in Wallace's approach. However, the layout is highly regular and the routing length is reduced.

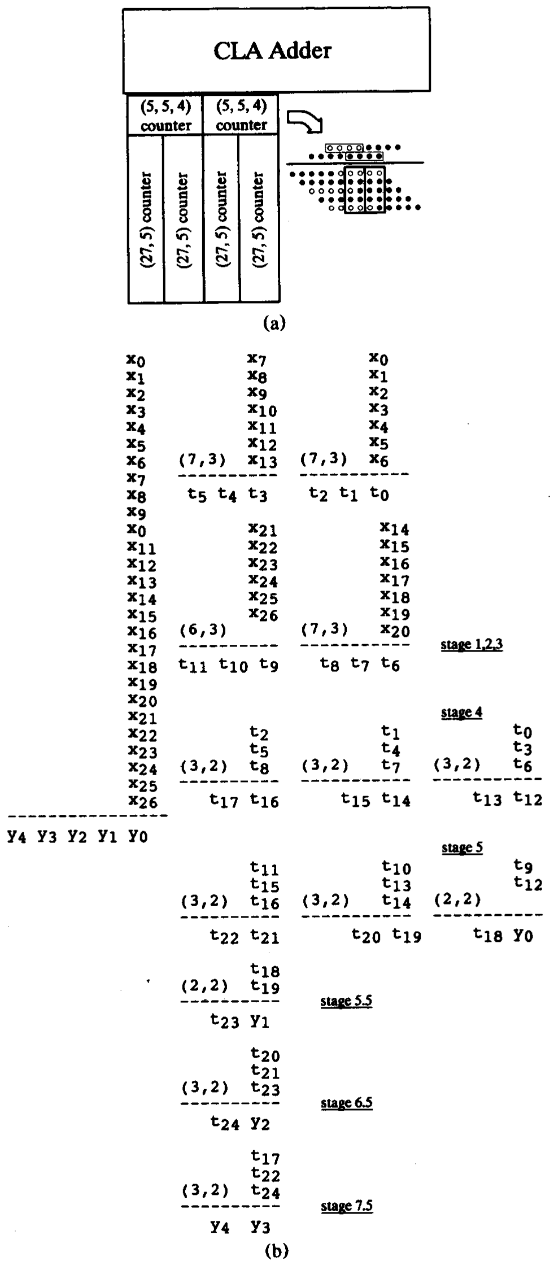


Fig. 11. (a) Partial-product reduction with (27,5) and (5,5,4) counters. (b) (27,5) counter.

D. The (9,2) Counter Family

Santoro and Horowitz have shown how to achieve a highly regular layout using (4,2) counters [2]. However, this requires one more stage than in Wallace's approach. We propose a new technique here that combines regular layout with a minimal number of stages.

The new approach uses the (9,2) counter family. The (9,2) counter family is a set of counters that includes (3,2), (4,2), (6,2), and (9,2) counters. We have previously shown how to construct a (4,2) counter from (3,2) counters. We extend the technique to construct a (6,2) counter by using two (3,2) counters and one (4,2) counter, as shown in Fig. 10(a). We partition the six inputs into two groups (x_5, x_4, x_3 as group 1 and x_2, x_1, x_0 as group 0) and we apply a (3,2) counter to each group. Thus we have four temporary signals, two of which have weight $2^0 = 1$ and two of which have weight $2^1 = 2$. The two signals with weight 2 are carried out to the *left* stage while the two

TABLE II
KEY TECHNOLOGICAL PARAMETERS

Layer	Signetics SABRE	Signetics Qubic	Stanford BiCMOS
Poly minimum feature size (μm)	0.8	1.2	2.0
Contact size (μm^2)	0.8	1.2	2.0
Metal 1 pitch (μm)	2.4	3.8	6.0
Metal 2 pitch (μm)	2.4	5.0	6.0

signals with weight 1 are carried in from the *right* stage. Thus we have total of $2+2=4$ signals of weight 1. We use one (4,2) counter to get the two final result bits. Therefore, each (6,2) counter has three equivalent (3,2) counter delay stages.

In a similar way, we can construct a (9,2) counter by using three (3,2) counters and one (6,2) counter, as shown in Fig. 10(b). Each (9,2) counter has four equivalent (3,2) counter delay stages.

The counters of this family can be used as follows for partial-product reduction. If we start with 27 partial-product rows, after applying the (9,2) counters at first stage, we reduce the tree height to 6, and then we can apply (6,2) counters to achieve the tree height of 2. Since the (9,2) counter is equivalent to four (3,2) counter stages and the (6,2) counter is equivalent to three (3,2) counter stages, the total number of equivalent (3,2) counter stages needed is seven, which is the same as in Wallace's approach. This structure can lead to highly regular layout, as shown in Fig. 10(c).

E. The (27,5) Counter

Another modular approach for partial-product reduction is proposed here. This scheme uses a combination of (27,5) counters and (5,5,4) counters. The partial-product reduction can be achieved with this method as shown in Fig. 11(a), where we assume to have 27 partial-product rows initially. The (27,5) counter can be seen as a vertical slice that embeds the local routing.

The (27,5) counter can be constructed as follows. The inputs are divided into four groups that feed (7,3) counters. (One (7,3) counter has an unused input.) Then the reduction is achieved by using (3,2) and (2,2) counters, as shown in Fig. 11(b). As far as propagation delay is concerned, the (27,5) counter is equivalent to 7.5 stages, if we count the (2,2) counter as half stage delay.

IV. SYNTHESIS OF MULTIPLIER STRUCTURES

The value of the circuits for partial-product reduction described before cannot be judged without a full layout analysis that allows us to take the parasitics into account and to measure precisely the circuit area and performance. Since several technological factors, such as metal pitch and device switching time, affect the figures of merit of the final implementation, we have decided to construct the layouts using *module generators*, which are portable across similar technologies.

The module generators are written in a layout language, and they encode the relative transistor position as well as their interconnection. Hierarchy is used to reduce the amount of code. Therefore, generators for (3,2) counters are used in those for high-input counters. Most routing among the cells is achieved by abutment, while intracell routing is specially designed to exploit the cell compactness. In this way, a highly regular layout can be obtained. We have used the L language and the GDT environment, which are available commercially from Mentor Graphics.

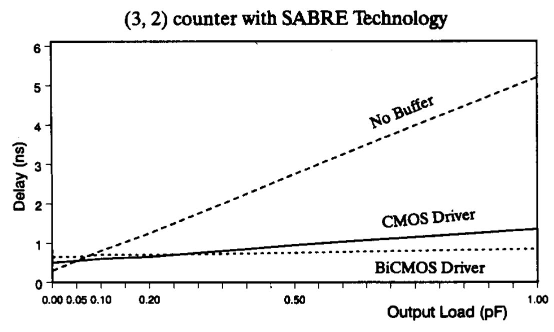
The module generators use information from a technology file that contains the parameters of the technology being used. Therefore, by changing the technology file we can achieve layouts in different (but similar) technologies. As a vehicle to understand the impact of the technological parameters over the area and performance of a multiplier, we have chosen three different BiCMOS technologies, namely:

1) SABRE, a submicrometer technology developed at Signetics [16], [17];
2) Qubic 1.5, a production-level technology developed at Signetics;
3) the Stanford BiCMOS process [20].

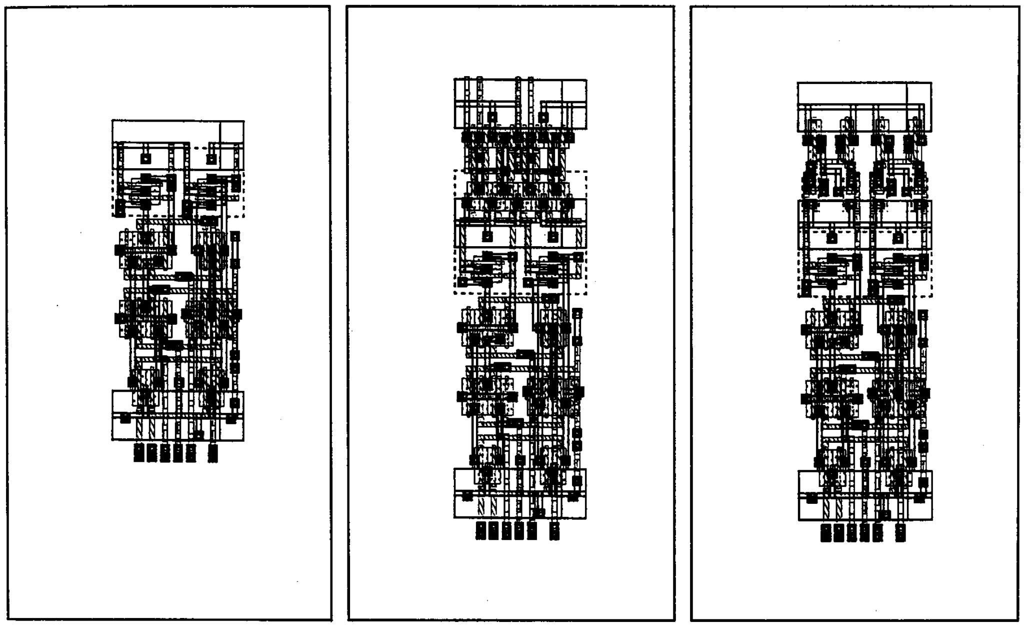
Some key parameters of these technologies are shown in Table II. It is important to remark that our study does not aim at an evaluation of these technologies as far as suitability for implementing high-performance arithmetic functions. These technologies represent three sets of data points among the existing technologies which have enough differentiation to significantly impact the implementation results, and therefore allow us to evaluate the suitability of the circuits being designed.

A. Overall Layout Considerations

We are considering here the partial-product reduction circuitry for a double-precision IEEE standard multiplier. This circuit is designed as a rectangular array. Its outputs feed a CLA adder and its inputs come from the Booth encoders/drivers. The size of the array is very critical in two respects. First, it must fit into a reasonable bounding box. Since the full IEEE double-precision standard mandates for an array with 106 columns, the width of each column is critical. In addition, the height of the array correlates to the wiring propagation delays, and it must be kept as short as possible. A regular layout approach reduces the amount of wiring and helps in keeping the size small. In the two layout styles considered below, the



(a)



No buffer (32.8 x 84 μm^2) CMOS buffer (32.8 x 113 μm^2) BiCMOS buffer (32.8 x 113 μm^2)

(b)

Fig. 12. (a) (3,2) counter performance with no buffer, CMOS driver, and BiCMOS driver. (b) (3,2) counter layouts with Signetics SABRE BiCMOS technology.

array consists of a set of vertical abutting slices, each corresponding to a column. We use two levels of metal for wiring. One level of metal is used for power buses and for bringing into the array the multiplier signals encoded by the Booth's scheme. The second level is used for local interconnection in the counter cells in the vertical direction. The multiplicand is brought into the array by jogging on the two metal layers. In principle a third level of metal could be used for this purpose.

As a general remark, the MOS feature sizes and the MOS and bipolar transistor switching speed affect the overall performance. The metal pitches affect both the area of the array and its performance, because of wiring delays.

B. Counter Layout Design

1) (3, 2) Counter: The (3,2) counter is the leaf cell of the hierarchy. A module generator for this counter uses parameters such as the device feature dimensions and the type of buffer desired (e.g., none, CMOS buffer, BiCMOS buffer). Fig. 12(a) shows the performance comparisons for different buffers and different loads using the SABRE process corresponding to the layout shown in Fig. 12(b). The general layout with BiCMOS buffers is shown in Fig. 13 for the three technologies under consideration.

2) (9, 2) Counter: The (9,2) counter, as well as the (6,2) counter, can be generated by a hierarchical combination of (3,2) counters. Layouts are shown in Fig. 14.

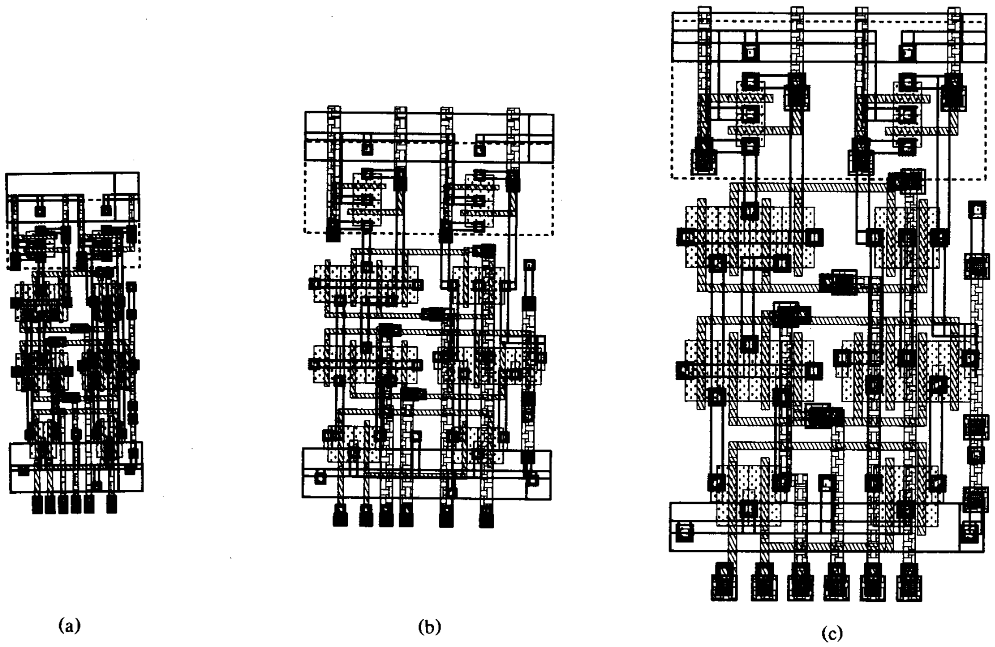


Fig. 13. Comparison of (3,2) counter in three BiCMOS technologies. (a) Signetics SABRE 0.8-μm BiCMOS technology (32.8×113 μm^2). (b) Signetics Quibic 1.2-μm BiCMOS technology (61.6×145 μm^2). (c) Stanford 2-μm BiCMOS technology (76×204 μm^2).

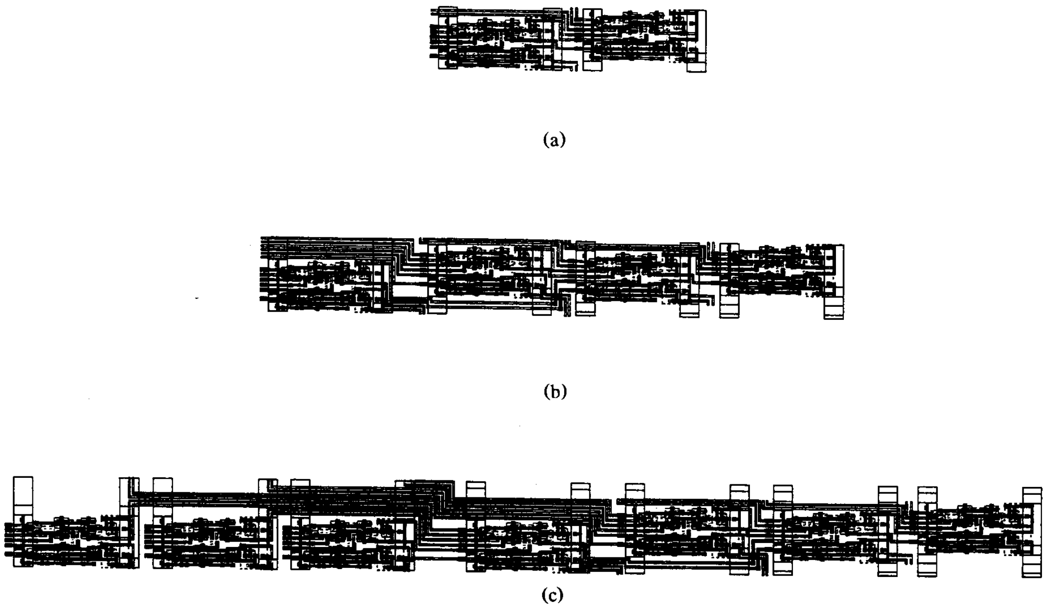


Fig. 14. Generated hierarchical counter layout in the SABRE technology. (a) (4,2) counter (37.6×176 μm^2 without Booth). (b) (6,2) counter (47.2×372 μm^2 without Booth). (c) (9,2) counter (56.8×659 μm^2 without Booth).

TABLE III
COMPARISON OF CHIP SIZE FOR THREE DIFFERENT TECHNOLOGIES FOR (9,2) COUNTER FAMILY APPROACH

Area (mm^2)	Signetics SABRE	Signetics Qubic	Stanford BiCMOS
Booth	8.2×4.2	15.7×5.9	19.0×7.8
No Booth	7.7×5.5	15.2×7.4	18.8×10.2
ratio	0.81	0.82	0.77

TABLE IV
COMPARISON OF SIMULATED SPEED FOR THREE DIFFERENT TECHNOLOGIES FOR (9,2) COUNTER FAMILY APPROACH

Delay (ns)	Signetics SABRE	Signetics Qubic	Stanford BiCMOS
Booth	4.8	9.3	22.4
No Booth	4.5	9.2	24.1
ratio	1.07	1.01	0.93

Note: Delay includes Booth encoders/decoders and whole tree partial-product reduction.

TABLE V
VARIATION OF SOME PROCESS PARAMETERS FOR SIGNETICS SABRE TECHNOLOGY

Cell size (μm^2)		(3,2) counter		(4,2) counter		(6,2) counter		(9,2) counter		tree with Booth			
		size	% change	size	% change	size	% change	size	% change	tree size	x%	y%	%
	original	32.8×83.8	100.0%	37.6×176	100.0%	47.2×372	100.0%	56.8×659	100.0%	8203×4149	100.0%	100.0%	100.0%
poly width	0.7	32.0×83.4	97.1%	36.8×175	97.3%	46.4×370	97.8%	56.0×656	98.1%	8118×4138	99.0%	99.7%	98.7%
	0.9	33.6×84.2	102.9%	38.4×177	102.7%	48.0×374	102.2%	57.6×662	101.9%	8288×4160	101.0%	99.9%	101.0%
contact size	0.7	31.8×83.0	96.0%	36.6×174	96.2%	46.2×369	97.1%	55.8×653	97.3%	8097×4128	98.7%	99.5%	98.2%
	0.9	33.8×84.6	104.0%	38.6×178	103.8%	48.2×375	102.9%	57.8×665	102.7%	8309×4170	101.3%	100.5%	101.8%
metal 1 pitch	2.3	32.8×83.8	100.0%	37.6×176	100.0%	47.2×372	100.0%	56.8×659	100.0%	8203×4149	100.0%	100.0%	100.0%
	2.5	33.8×83.8	103.0%	38.6×176	102.7%	48.2×372	102.1%	57.8×659	101.8%	8309×4149	101.3%	100.0%	101.3%
metal 2 pitch	2.3	32.8×83.8	100.0%	37.4×175	98.9%	46.6×371	98.5%	55.8×656	97.8%	8039×4139	98.0%	99.8%	97.8%
	2.5	32.8×83.8	100.0%	37.8×177	101.1%	47.8×373	101.5%	57.8×662	102.2%	8367×4159	102.0%	100.2%	102.2%

3) (27,5) Counter: The (27,5) counter can be generated using (7,3) and (3,2) counters. In turn, the (7,3) counters is efficiently designed using (3,2) counters as well.

C. Array Implementation with the (9,2) Counter Family

We have generated layouts of the partial-product reduction array using (9,2) and (6,2) counters by the scheme described in Section III-D. The array consists of 106 adjacent columns. Each column holds three (9,2) counters and a (6,2) counter, plus Booth decoders. Tables III and IV compare the array area and its performance in the three selected technologies. It is evident that a small metal pitch of both metal levels is key to an efficient implementation.

This approach can be extended to the case in which no Booth encoding is used. In this case we have more partial-product rows (53) but no Booth decoder circuits in the column slice. It is interesting to notice that the modified Booth's scheme is slightly slower than the one without such encoding for the SABRE technology. The reasons are the following:

- encoding and decoding circuits take significant delays;
- the decoder circuit increases the column slice height and hence it offsets the effect of having fewer partial-product terms;
- the total number of wires entering the array from the Booth's encoders is higher by about 50%. This is due to the fact that the 2-b modified Booth's scheme uses 3 b to encode the five different combinations $(0, +X, -X, +2X, -2X)$ (Fig. 2) for each pair of bits of the multiplier.

We would like to comment now on the impact of the individual design rules on the area and performance. For this reason, we leverage the advantage of having used module generators for the layout, which allow us to perturb some key design rule values and evaluate the effect on the overall layouts that are synthesized automatically.

We have considered four important parameters, namely: polysilicon width, contact size, metal 1 pitch, and metal 2 pitch. We have perturbed these parameters by 0.1 μm (about 10%) independently.

Let us consider the SABRE technology first. The results for area are reported in Table V. The major result is that the area of the basic macrocell (3,2) counter is affected by changes in contact size and polysilicon width. The relative variations are 4% and 2.9%, respectively. When we reduce the metal 1 pitch, the cell size is unchanged because other design rules prohibit the cell size from becoming smaller. The metal 2 is used mainly in routing outside the (3,2) macrocell, therefore, it does not

TABLE VI
Variation of Some Process Parameters for Signetics Qubic Technology

Cell		(3,2) counter		(4,2) counter		(6,2) counter		(9,2) counter		tree with Booth	
size		size	% change	size	% change	size	% change	size	% change	size	%
(μm^2)	original	61.6×102	100.0%	64.6×227	100.0%	84.6×497	100.0%	111.6×885	100.0%	15650×5902	100.0%
poly	1.1	60.8×101	97.7%	63.8×226	98.3%	83.8×495	98.7%	110.8×882	98.9%	15640×5817	98.5%
width	1.3	62.4×103	102.3%	65.4×228	101.7%	85.4×499	101.4%	112.4×888	101.1%	15660×5987	101.5%
contact	1.1	60.6×101	97.4%	63.6×225	97.6%	83.6×494	98.2%	110.6×879	98.4%	15630×5796	98.1%
size	1.3	62.6×103	102.6%	65.6×229	102.4%	85.6×500	101.8%	112.6×891	101.6%	15670×6008	101.9%
metal 1	3.7	61.6×102	100.0%	64.6×227	100.0%	84.6×497	100.0%	111.6×885	100.0%	15650×5902	100.0%
pitch	3.9	62.6×102	101.6%	65.6×227	101.5%	85.6×497	101.2%	112.6×885	100.9%	15650×6008	101.8%
metal 2	4.9	61.6×102	100.0%	64.4×226	99.3%	84.0×496	99.1%	110.6×882	98.8%	15640×5796	98.1%
pitch	5.1	61.6×102	100.0%	64.8×228	100.8%	85.2×498	100.9%	112.6×888	101.2%	15660×6008	101.9%

TABLE VII
Variation of Some Process Parameters for Stanford BiCMOS Technology

Cell		(3,2) counter		(4,2) counter		(6,2) counter		(9,2) counter		tree with Booth	
size		size	% change	size	% change	size	% change	size	% change	size	%
(μm^2)	original	76×145	100.0%	89×320	100.0%	103×656	100.0%	136×1175	100.0%	19072×7781	100.0%
poly	1.9	75×144	98.0%	88×319	98.6%	102×654	98.7%	135×1172	99.0%	18966×7771	99.3%
width	2.1	77×146	102.0%	90×321	101.4%	104×658	101.3%	137×1178	101.0%	19178×7791	100.7%
contact	1.9	75×144	98.0%	88×318	98.3%	102×653	98.6%	135×1169	98.8%	18966×7751	99.1%
size	2.1	77×146	102.0%	90×322	101.8%	104×659	101.4%	137×1181	101.2%	19178×7801	100.8%
metal 1	5.9	76×145	100.0%	89×320	100.0%	103×656	100.0%	136×1175	100.0%	19072×7781	100.0%
pitch	6.1	77×145	101.3%	90×320	101.1%	104×656	101.0%	137×1175	100.7%	19178×7781	100.6%
metal 2	5.9	76×145	100.0%	89×319	99.7%	102×655	98.9%	135×1172	99.0%	18966×7771	99.3%
pitch	6.1	76×145	100.0%	89×321	100.3%	104×657	101.1%	137×1178	101.0%	19178×7791	100.7%

affect the cell size. However, when considering the full tree, a 2.2% area variation is due to metal 2.

We simulated the circuits for the perturbed values of the parameters. Simulations indicate that the impact on speed is due to the variation in polysilicon width (channel length) only. We simulated at room temperature with $V_{cc} = 4.0$ V. (We used 4.0 V for our simulation because the model showed punchthrough characteristic above 4.0 V for 0.7-μm channel length.) We found that the speed variations for the unloaded (3,2) counter are -0.1 ns (from 0.6 to 0.5 ns) (17%) and $+0.2$ ns (from 0.6 to 0.8 ns) (33%). The delay of the whole tree varies from -1.4 ns (from 4.8 to 3.4 ns) (29%) to $+2.3$ ns (from 4.8 to 7.1 ns) (48%).

We show in Tables VI and VII the variation of cell area when perturbing some process parameters for the other technologies. Results are qualitatively similar to the previous case, with poly width and contact size affecting the area of the (3,2) counter. Metal 2 affects the overall area. We have also simulated the change of delay caused by varying the polysilicon width (channel length). The results are -1.8 and $+2.2$ ns for Qubic technology and -2.1 and $+2.3$ ns for Stanford BiCMOS technology.

D. Array Implementation with the (27,5) and the (5,5,4) Counters

We consider here a partial-product reduction array using the (27,5) and the (5,5,4) counters, as described in Section III-E. This array uses the modified Booth's encoding scheme and the decoder circuits described in Section III-A. We designed the array with module generators, but we carried out the complete synthesis using the SABRE technology only.

The array consists of 53 vertical slices. Each slice has two (27,5) counters that are on top of one (5,5,4) counter (Fig. 11(a)). The (27,5) circuits embed the Booth decoding circuits. The area of the entire array amounts to 3.5×10 mm^2.

We have implemented two vertical slices on silicon on SABRE technology. A picture of the test chip is shown in Fig. 15. We have then measured the propagation delay through the slice related to the slowest input-to-output path, as shown in Fig. 16. The worst-case delay is about 12 ns (Fig. 16), which reduces to 9 ns when we exclude the probe and I/O delays.

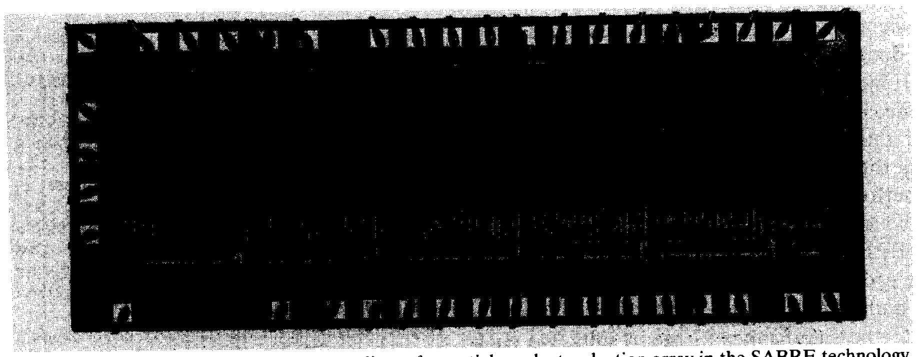


Fig. 15. Silicon implementation of two slices of a partial-product reduction array in the SABRE technology.

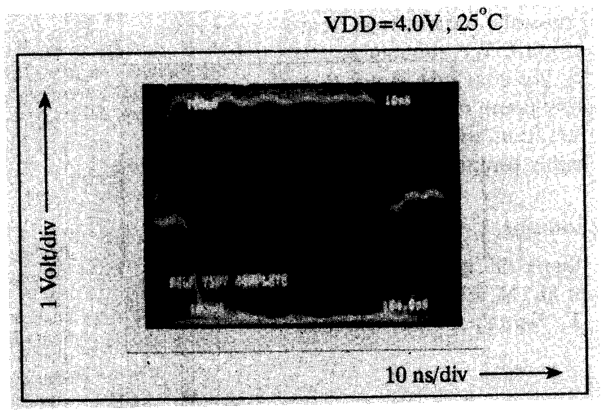


Fig. 16. Measured output waveform from the silicon implementation in SABRE technology.

TABLE VIII
COMPARISON OF DIFFERENT APPROACHES FOR IEEE STANDARD DOUBLE-PRECISION MULTIPLICATION

	Weitek 3364	TI 74ACT8847	Intel i860	IBM 6000	(9,2) family	(28,5)+(5,5,4)
cycle (ns)	50	30	20	40	12 (Note 2)	15 (Note 2)
size (mm2)	95.5	100.6	154.8	161.3	41.9 (Note 3)	48.4 (Note 3)
# of transistors	165 000	180 000	170 000 (total 1 000 000)	420 000	85 000	90 000
latency (# of cycles) (Note 1)	2	3	3	2	1 (single cycle)	1 (single cycle)
counter used	(3,2)	redundant binary-tree	(4,2)	(7,3)	(9,2) family	(28,5) +(5,5,4)

Note 1. IEEE double-precision multiplication.
Note 2. Simulated at room temperature and 4.0-V V_{cc}, typical process corner.
Note 3. Area and # of transistors are estimated for IEEE double-precision multipliers implementation only.

V. Analysis and Conclusion

We have analyzed circuits, structures, and physical implementations of the partial-product reduction array portion of a multiplier. We have synthesized layouts for the array in different BiCMOS technologies by using module generators with the goal of comparing the results that stem from different circuit and structure choices when applied to different sets of layout rules.

The set of design rules and the device characteristics of the chosen technology determine the area and speed of the circuit. In our case, the choice of a regular array structure permits us to directly relate the design rules to the performance goals. For a parallel multiplier implementation that does not use iteration, a key issue is the size of the partial-product array because it affects the wiring length and its corresponding delay. Limiting factors are the metal pitch in both layers and the MOS features. We think that submicrometer CMOS with at most a 2.5-μm metal pitch is required to have an effective implementation. A third level of metal and/or low-resistivity local interconnect is also highly desirable.

We have designed and implemented a test slice for a floating-point double-precision multiplier in an experimental submicrometer BiCMOS technology. Based on the performance measurements on silicon of the reduction tree, we think that it is possible to design a complete multiplier using the (27,5) counter approach in that technology which performs an operation in about 12 ns, while 15 ns would be required to support all the rounding modes of the IEEE standard [1]. Considering the simulation results, we project that it would be possible to design an even faster multiplier using the (9,2) counter family.

To appreciate the significance of our results, we compare our implementation and its performance to other existing commercial multipliers. The comparisons can be only qualitative, because other approaches use different technologies, algorithms, and possibly margins (e.g., temperature range, supply range, etc.). We summarize the results in Table VIII [21]–[23]. We conclude that a double-precision IEEE 754 floating-point multiplier having 12-ns delay and requiring no iteration, which can be built with our schemes, would be highly competitive.

Acknowledgment

The authors would like to thank the generous support from Signetics and in particular Dr. M. El-Diwany, Dr. U. Bar Gadda, Dr. V. Akylas, P. Baltus, G. Bewick, and N. Quach.

References

[1] *IEEE Standard for Binary Floating-Point Arithmetic*, ANSI/IEEE Standard 754-1985, Aug. 12, 1985.
[2] M. Santoro and M. Horowitz, "A pipelined 64×64 b iterative array multiplier," in *ISSCC Dig. Tech. Papers*, Feb. 1988, pp. 36–37.
[3] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Trans. Electron. Comput.*, vol. EC-13, pp. 14–17, Feb. 1964.
[4] L. Dadda, "Some schemes for parallel multipliers," *Alta Freq.*, vol. 34, pp. 349–356, Mar. 1965.
[5] R. K. Montoye, P. W. Cook, E. Hokenek, and R. P. Havreluk, "An 18 ns 56-bit multiply-adder circuit," in *ISSCC Dig. Tech. Papers*, Feb. 1990, pp. 46–47.
[6] L. Kohn and S. W. Fu, "A 1,000,000 transistor microprocessor," in *ISSCC Dig. Tech. Papers*, Feb. 1989, pp. 54–55.
[7] H. Ling, "High-speed binary adder," *IBM J. Res. Develop.*, vol. 25, pp. 156–166, May 1981.
[8] G. Bewick, P. Song, G. De Micheli, and M. J. Flynn, "Approaching a nanosecond: A 32 bit adder," in *ICCD Proc.*, Oct. 1988, pp. 221–226.
[9] M. Santoro, G. Bewick, and M. A. Horowitz, "Rounding algorithms for IEEE multipliers," in *Proc. 9th IEEE Symp. Comput. Arithmetic*, 1989, pp. 176–183.
[10] N. Quach, N. Takagi, and M. Flynn, "On fast IEEE rounding," Stanford CSL, Stanford, CA, Tech. Rep. CSL-TR-91-459.
[11] S. Waser and M. Flynn, *Introduction to Arithmetic for Digital Systems Designers*. New York: Holt, Rinehart and Winston, 1982.
[12] F. S. Anderson, J. G. Earle, R. E. Goldschmidt, and D. M. Powers, "The IBM system 360/91 floating point execution unit," *IBM J. Res. Develop.*, vol. 11, no. 1, pp. 34–53, Jan. 1967.
[13] L. G. Heller and W. R. Griffin, "Cascode voltage switch logic: A differential CMOS logic family," in *ISSCC Dig. Tech. Papers*, Feb. 1984, pp. 16–17.
[14] N. Weste, *Principles of CMOS VLSI Design, A Systems Perspective*. Reading, MA: Addison-Wesley, 1985, pp. 169–171.
[15] Y. Shimazu *et al.* "A 50 MHz 24b floating-point DSP," in *ISSCC Dig. Tech. Papers*, Feb. 1989, pp. 44–45.
[16] M. El-Diwany *et al.*, "An advanced BiCMOS process utilizing ultra-thin silicon epitaxy over arsenic buried layers," in *IEDM Tech. Dig.*, Dec. 1989, pp. 245–248.
[17] M. El-Diwany, M. Brassington, R. Razouk, P. V. Wijnen, and V. Akylas, "Low voltage performance of an advanced CMOS/BiCMOS technology featuring 18 GHz bipolar f_T and sub-70 ns CMOS gate delays," in *IEDM Tech. Dig.*, Dec. 1990, pp. 489–492.
[18] D. T. Shen and A. Weinberger, "4-2 carry-save adder implementation using send circuits," *IBM Tech. Disc. Bull.*, vol. 20, pp. 3594–3597, Feb. 1978.
[19] A. Weinberger, "4-2 carry-save adder module," *IBM Tech. Disc. Bull.*, vol. 23, Jan. 1981.
[20] J. Shott, "The Stanford BiCMOS project," Stanford Univ., Stanford, CA, CIS internal report, Sept. 1990.
[21] J. L. Hennessy and D. A. Petterson, *Computer Architecture, A Quantitative Approach*. Los Altos, CA: Morgan Kaufmann, 1990, p. A-53.
[22] H. B. Bakoglu *et al.*, "IBM second-generation RISC machine organization," in *Proc. IEEE ICCD*, Oct. 1989, pp. 138–142.
[23] T. Perry, "Million-transistor microchip, Intel's secret is out," *IEEE Spectrum*, pp. 22–28, Apr. 1989.

Paul J. Song received the B.S. degree in electrical engineering from National Taiwan University in 1977, and the M.S. degree in electrical engineering from University of California at Santa Barbara in 1981. He is currently working toward the Ph.D. degree at Stanford University, Stanford, CA.

He has industrial experience with AMD, EXEL, ICT, and ISSI in EPROM, EEPROM, PLD, SRAM, and arithmetic processing unit.

Giovanni De Micheli (S'79–M'83–SM'89) received the Dr. Eng. degree, summa cum laude, in nuclear engineering from the Politecnico di Milano, Italy, in 1979, and the M.S. and Ph.D. degrees in electrical engineering and computer science from the University of California, Berkeley, in 1980 and 1983, respectively.

He is Associate Professor of Electrical Engineering and, by courtesy, of Computer Science at Stanford University, Stanford, CA. From 1984 to 1986 he worked at the IBM T. J. Watson Research Center, Yorktown Heights, NY, where he was project leader of the Design Automation Workstation group. Previously he held positions at the Department of Electronics of the Politecnico di Milano, Italy, and at Harris Semiconductor, Melbourne, FL. His research interests include several aspects of the computer-aided design of integrated circuits with particular emphasis on automated synthesis, optimization, and verification of VLSI circuits.

Dr. De Micheli was granted a Presidential Young Investigator Award in 1988. He received the 1987 Best Paper Award for the best paper published in the IEEE Transactions on Computer-Aided Design and a Best Paper Award at the 20th Design Automation Conference in June 1983. He was co-director of the Advanced Study Institute on Logic Synthesis and Silicon Compilation, held in L'Aquila, Italy, under the sponsorship of NATO in 1986 and in 1987. He was Technical and General Chairman of the International Conference on Computer Design (ICCD) in 1988 and 1989, respectively.