Improving Multiplier Design by Using Improved Column Compression Tree and Optimized Final Adder in CMOS Technology

Vojin G. Oklobdzija, *Senior Member, IEEE,* and David Villeger

***Abstract*—In this paper we discuss improvements in bit reduction techniques in a parallel multiplier and the use of a final adder which is optimized for the uneven signal arrival profile. Different architectures of the column compressors and the use of carry propagate adders which take advantage of the speed of the carry signal are considered. The column compressors configuration is optimized in order to reduce the longest signal path. The final adder is designed for the uneven input arrival time of the signals originating from the multiplier tree. This results in more compact wiring and balanced delays yielding a faster multiplier.**

I. Introduction

THE CRITICAL SIGNAL path in a parallel multiplier can be divided into three domains: Booth encoder (BE), partial-product summation tree (PPST) and carry-propagate adder (CPA). The delay introduced by the Booth encoder is relatively small compared to the other two components, especially for the large size multiplier. This delay component is also relatively independent of the size of the multiplier. The delay introduced by the PPST and the CPA constitutes a dominant component of the delay in the multiplier where the delay introduced by the PPST is about two to four times that of the Final adder. Therefore improvement in those two parts will result in major enhancement of the speed of the multiplier.

In this paper we discuss the efficiency of various known ways of compressing the bit matrix. We will examine the efficiency of the several column compression techniques as applied in [1]–[9], [11], [13], [14], [21] and discuss ways of optimizing the critical path in the multiplier. We considered a 24-b integer multiplier in our example, though the results with respect to PPST are in general applicable to 1's and 2's complement (with or without Booth encoding). We have designed several different test configurations using LSI Logic 100K 1μ CMOS ASIC cells [22] which were simulated and analyzed for possible delay optimization.

Manuscript received September 8, 1993; revised July 5, 1994. This work was supported in part by the Office of Research, University of California Davis by a minigrant, and by Hitachi Ltd. and California MICRO under Grant 92-115.V.

V. G. Oklobdzija is with the Electrical and Computer Engineering Department, University of California, Davis, CA 95616 USA (e-mail: vojin@ece.ucdavis.edu).

D. Villeger is with Ecole Superieure d'Ingenieurs en Electrotechnique et Electronique, 93162 Noisy le Grand CEDEX, France, (e-mail: villeger@apo.esiec.fr).

IEEE Log Number 9410840.

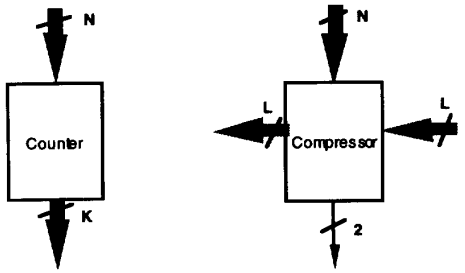


Fig. 1. (a) Counter and (b) compressor.

A. Definitions

In the following text we will make a distinction between *counters* and *compressors*, as the two have often been confused in the literature.

We define a **counter** as a (combinational logic circuit) device where the number of the output lines is $K \geq \log_2(N+1)$ and N is the number of input lines (Fig. 1(a)). The status of the output lines K represents the binary number equal to the number of input lines $M \leq N$ that are asserted to a logic one. The output represents a count of active inputs.

We define a **compressor** $(\mathbf{C}_i)$ to be a (combinational network) device that compresses N input lines (column of bits) in the bit position i to two output lines. In addition there are L input lines entering the compressor at different levels j and the same number of lines L originating from the compressor at the same levels j, where $L = N - 3$, as shown in Fig. 1(b). The L input lines entering the compressor C_i are in the bit position i. The L output lines from the compressor C_i are in the bit position $i+1$, and enter the compressor C_{i+1}. The purpose of a compressor is to reduce the number of bits to two. These L lines entering and originating from the compressor are carry signals. It is critical that the output signals L^{i+1} from the compressor C_i are connected in the same order to the input lines L^{i+1} of the compressor C_{i+1} so signals originating from some level j of C_i enter the same level j of C_{i+1}. This preserves the signal ordering and imposes a particular interconnection structure of the partial-product summation tree.

An additional distinction between a *counter* and a *compressor* concerns the output. The output of a *counter* represents a binary number corresponding to the number of 1's at the

1063–8210/95$04.00 © 1995 IEEE

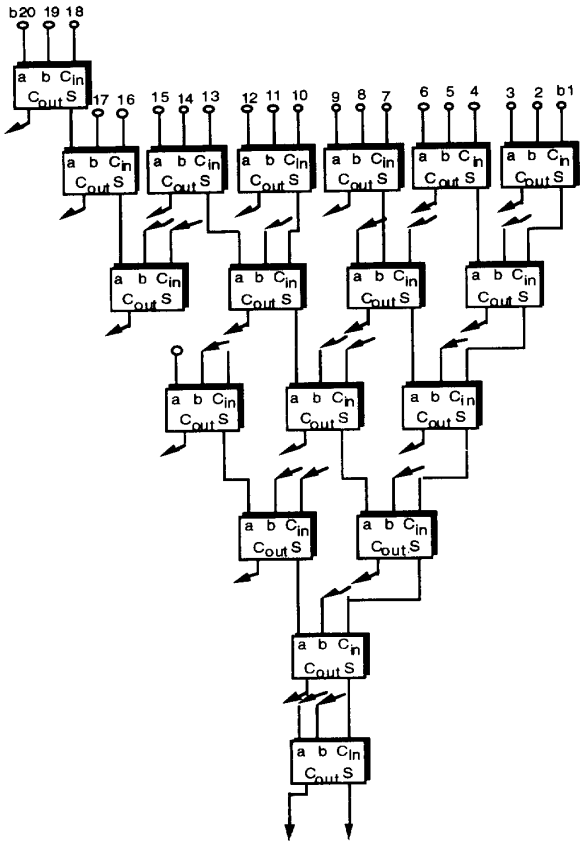


Fig. 2. Carry-save summation tree as introduced by Wallace [3].

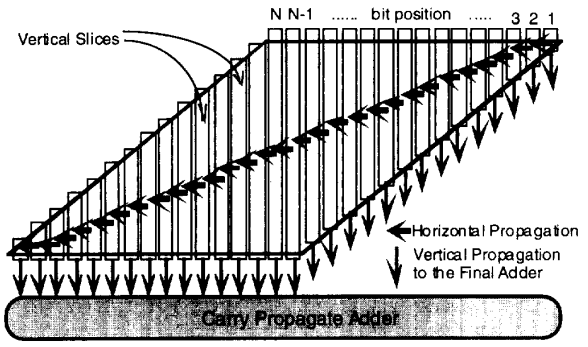


Fig. 3. Partial-product summation tree of a multiplier shown as a multiple of carry-save summation trees.

counter inputs. This is not the case with the *compressor* whose output consists of two signals augmented by L lines originating from the *compressor*. We cannot obtain a direct binary *count* of 1's entering the compressor from these $L+2$ lines, although interpretation of that count in a redundant number system is possible.

B. Summation of the Partial-Products

Summation of the partial products in a parallel multiplier is traditionally done by using full adders (FA) arranged in a carry-save summation tree (CSST) [3]. It should be noted that this is in essence a 3-D tree structure where a vertical cross-section represents a CSST for a particular bit position i. A vertical cross-section at the twentieth bit position, as introduced by Wallace, is shown in Fig. 2. We have modified the original Wallace figure and point out an important detail missing from that figure, that the carry signals entering the twentieth cross-section (from bit position 19) also originate from it (into bit position 21).

A critical path in the PPST can be contained entirely in the CSST at bit position N, or could be spanning several CSST's ending in the bit position N in the middle of the PPST, as shown in Fig. 3. To improve the speed of PPST we must not only reduce the number of levels in the CSST at bit position N, but also ensure that all of the signals originating from lower order CSST bit positions $i = 1, \ldots, N-1$ do not contribute to the delay of the signal in the bit position N. It can be seen in Fig. 3 that the delay of the *vertical critical path* of the CSST at bit position i (where $i = 1, \ldots, N-1$) is not important because it contains fewer levels than the CSST at bit position N.

II. Critical Path Optimization in the Partial Product Summation Tree

The objective of the work presented in this section is to examine and compare the known methods for summation of the partial products and suggest a new alternative. Beginning with Wallace's "*A Suggestion for Fast Multipliers*" [3], partial product summation has traditionally been done by using a carry-save adder tree arrangement. A significant departure from carry-save adder arrangement has been achieved by introduction of *compressors* such as 4:2 [5], [7] and 9:2 [11] which involve limited (1-bit) carry propagation. We intend to explore the concept of limited carry propagation by introducing short carry-propagate adders in the PPST as a means of fast and efficient summation of patial products [17]. We compare this approach with others by comparing first the critical paths in terms of XOR gate delays, and secondly by performing simulation on a number of multiplier test cases designed using LSI Logic 100K 1μ CMOS ASIC technology and simulating them using LSI Logic C-MDE simulator [22]. The simulation takes into account loads on the gate output due to the capacitance of the inputs of the gates connected to the particular output, as well as an average wire delay. Given that in ASIC pacement and wiring is done with the help of computer tools and imposed on a prefabricated structure, the average wire lengths are estimated from the size of the chip, complexity of the design and the package used. Therefore, the results are not measured delays, though our experience has shown very good agreement between simulation results and delays that were actually measured. However, our particular interest is not in absolute delay values, but rather in relative differences in delays between the various multiplier schemes considered.

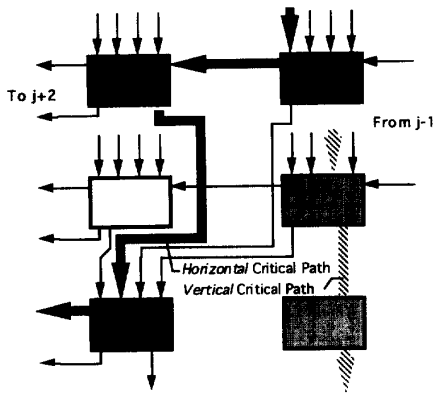


Fig. 4. Critical signal path in the multiplier tree consisting of 4 : 2 compressors.

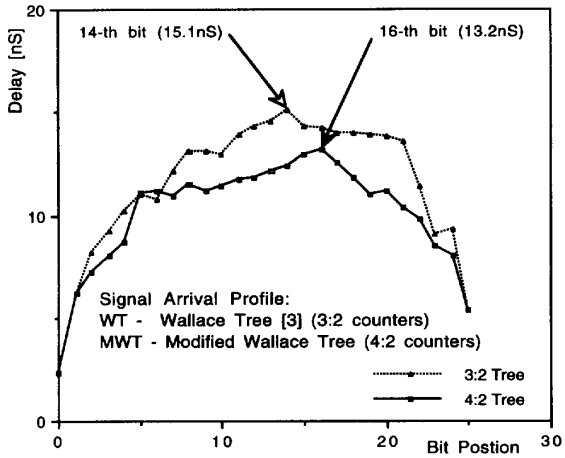


Fig. 5. Signal arrival profile from the regular (WT) and modified (MWT) column compression tree (12 × 12-b 2's complement multiplier with Booth encoding in LSI 100K technology [13]).

A. Approach

Signals from the multiplier tree do not arrive at the last stage (final adder) at the same time (Fig. 4). This is due to the fact that the number of partial-product bits is larger in the middle of the multiplier tree, and also, because we apply carry-save summation (or use counters) across the entire multiplier tree. This observation leads us to believe that we should reduce the paths in the middle by allowing the signals to cross into a portion of the multiplier tree with the smaller *depth* (i.e., where the number of partial-product bits in the column is smaller). We could also use different types of counters (even full adders) in the middle of the multiplier tree rather than at the ends. Our objective is to reduce the signal arrival time for the paths in the middle at the expense of the paths at the ends of the tree. Finally, after *flattening* the signal arrival profile as much as possible, we take advantage of the un-even signal arrival profile by *tuning* the final adder into the resulting profile for an additional gain in speed.

B. Use of 4 : 2 Compressors

A major departure from the traditional use of the 7 : 3 and 3 : 2 counters in the implementation of the Wallace tree began with the 1981 idea of A. Weinberger of IBM [5] which was made visible by the work of M. Santoro in [7] and implemented by others [8], [9]. The significance of a 4 : 2 compressor has not only been in the higher compression ratio of 2, but in the reduced delay due to the ability to redesign the circuitry of the 4 : 2 compressor [8]. The resulting delay of a 4 : 2 compressor is reduced to three XOR equivalent CMOS gates (instead of four XOR) resulting in faster column compression. In our case, this results in 12 XOR equivalent gate delays for a 24 × 24-b multiplier. However, the significance of the 4 : 2 compressor has been in the departure from the traditional *vertical* direction of the signal path. In our design, which uses the compressors of higher compression ratios, *horizontal* as well as *vertical* propagation of the signal is allowed. We designate the signal path as vertical if the signal travels from the partial-product generators and stays in the same bit position i until the final adder. Similarly, a *horizontal* path is defined as one where the signal propagates from a lower bit position i to a higher bit position $i+1$. An example of how *horizontal* propagation is used is the C_{out} signal from the 4 : 2 compressor. *Vertical* and *horizontal* critical paths are shown in Fig. 4.

Comparing the signal arrival profiles from the Wallace tree (WT) implemented from 3 : 2 counters and the modified Wallace tree (MWT) implemented using 4 : 2 compressors, we can observe that the MWT has its maximal delay shifted slightly toward the right (higher order bits) with slightly reduced maximum. This comes from the fact that the critical path in the tree travels not only in *vertical* but in *horizontal* direction as well.

We use the fact that we can achieve faster horizontal signal propagation by special design of the column compressors and interleaving the PPST in such a way that we use the fastest combination in what is normally the longest signal path in the bit compression tree. Using mixed sizes of column compressors results in not only a more balanced PPST but also a more even signal arrival profile of the inputs to the CPA as shown in Fig. 5. Optimization of the compressor cell for the 4 : 2 compressor is shown in Fig. 6(a) and results in almost equal propagation time delays on its outputs (Fig. 7) which, in our example, is due to the limitations of the CMOS library. Note especially, the carry signal can be made faster than the sum which leads to the reduction of the critical path involving *horizontal* propagation of the signals; this is the basis of our design. Note also, the delay of the middle columns has been reduced at the expense of the side columns and the worse case occurs in the bit 29 position (for a 24 × 24-b multiplier).

C. 9 : 2 Compressor

The idea of using higher order compressors extends itself naturally to the use of 9 : 2 and higher compressors such as 27 : 5 [11]. The advantage of using higher order compressors is primarily more compact and regular wiring, while a disadvantage is that due to their size, their efficiency does not

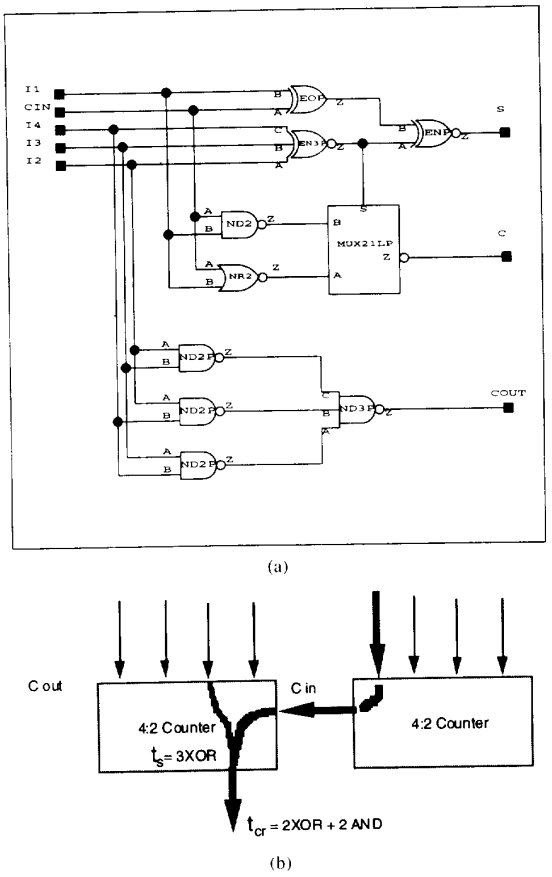


Fig. 6. (a) Optimized 4 : 2 compressor. (b) Signal delay in 4 : 2 compressor.

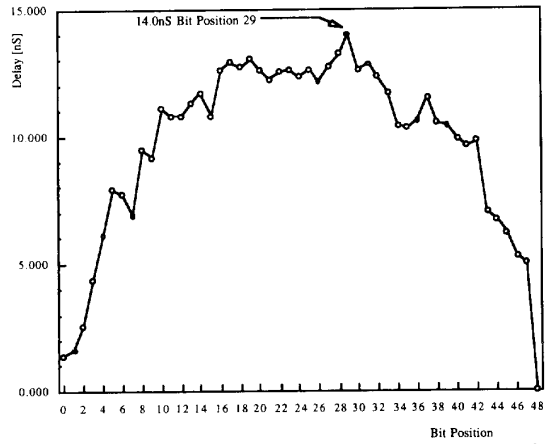


Fig. 7. Delay profile of the PPST using optimized 4 : 2 compressors (24 × 24-b integer multiplier in LSI 100K technology).

increase with the size of the compressor. The structure of a 9 : 2 compressor is shown in Fig. 8.

In the case of a 24 × 24-b multiplier, using a 9 : 2 compressor would result in two stages. In the first stage we have three 9 : 2 compressors reducing the number of bits to 6 which is

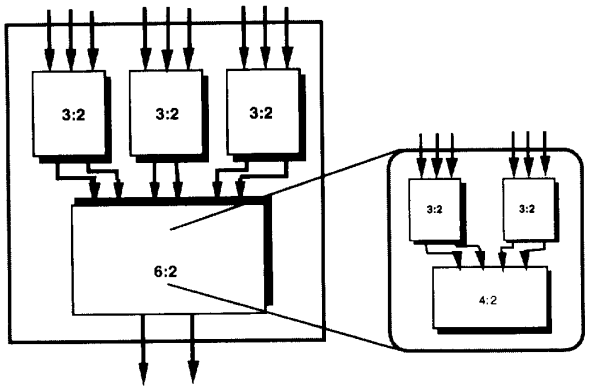


Fig. 8. 9 : 2 compressor.

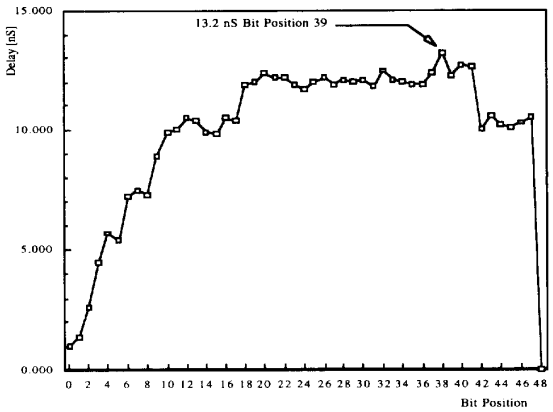


Fig. 9. Signal delays from the PPST implemented using 9 : 2 compressors (24 × 24-b integer multiplier in LSI 100K Technology).

then compressed into two with one stage of 6 : 2 compressors. The delay of such a tree, for a 24 × 24-b multiplier, measured in terms of XOR gate delays is equivalent to 11 XOR gate delays. Fig. 9 shows the signal delays for the multiplier tree consisting of 9 : 2 compressors.

D. Using Carry-Propagate Adders in the PPST

If we are to examine the rate at which rows of the partial product bits are being reduced, we observe that the PPST that uses 9 : 2 compressors (or the higher order ones) achieves this reduction more rapidly than the PPST that utilizes Wallace tree or 4 : 2 compressors. Since the compression ratio, C_r, is the ratio of the number of rows of the previous stage to the number of rows of the subsequent stage, we arrive at $C_r = 4.5$ for a 9 : 2 compressor and $C_r = 13.5$ for a 27 : 2 compressor. A stage is a step in the row reduction process as defined by Dadda [4].

On the other hand the compression ratio is $C_r = 2$ and 1.5 for 4 : 2 compressor and (3, 2) counter, respectively. However, this might be misleading because we can not build higher order compressors of the same speed as the lower order ones. For example, 4 : 2 compressor has $C_r = 2$ but this is achieved with four XOR gates in the critical path of the compressor (three

TABLE I
COMPRESSION RATIOS FOR FULL ADDERS AND SEVERAL COMMONLY USED COMPRESSORS

Compressor (Counter)	(3,2)	4:2	(7,3)	6:2	9:2	(27,5) [10]
C_r	1.5	2.0	2.33	3	4.5	5.4
No. of XOR gates in the critical path	2	3	4	5	7	12
Compression ratio per XOR gate, c_r	0.75	0.66	0.58	0.6	0.64	0.45

XOR gates for the improved version), while the compression ratio $C_r = 1.5$ is achieved with two XOR gates in the critical path if a full adder is used instead. This yields a compression ratio per equivalent XOR gate $(c_r), c_r = 0.5$ for $4:2$ compressor (0.66 for the improved compressor) which is worse than $c_r = 0.75$ if Full Adders are used. The compression ratios C_r and c_r for full adders and several compressors are given in Table I.

As the table shows, the best compression ratio per XOR gate (c_r) is that of a full-adder.

Consider the following example. Let us assume that we use an array of K-b carry propagate adders in the PPST. Also let us assume the use of carry propagate adders of K b such that the carry propagation T_c takes less time than the sum $T_s (T_c < T_s)$. In such a case, the use of K-b carry propagate adders would be equivalent to a $(2K{:}K)$ compressor so $C_r = 2$ and $c_r = 1.0$, which is higher than any of the compressors (counters) compared in Table I. (There are two XOR gates in the sum path.)

By introducing *horizontal* and *vertical* signal propagation in the PPST, we can trade some of the speed by adjusting the critical path to travel in the *horizontal* direction for some portion of the signal path. We are taking advantage of the fact that we can design a compressor (adder) in such a way that the carry signal can be realized faster than the sum. Therefore, our assumption is that we can accomplish a *horizontal* propagation in the same time as *vertical*, which is the best condition achievable.

1) Signal Delay: If we use K-b adders in the PPST instead of compressors, the size K will be determined by the time necessary to propagate a carry signal. For the optimal adder, the time taken by the carry signal (T_c) equals that of the sum (T_s). In many CMOS and nMOS implementations, a carry path is implemented as a *pass-transistor* chain which is several times faster than the equivalent gate implementation [17]. We have chosen a 4-b adder for our analysis. The compression ratio C_r of this scheme is slightly smaller than 2 (as with $4:2$ compressor), because we have to collect carry signals after each 4 b. *Vertical* delay of the full adder is equivalent to two XOR gates, which is better than that of the $4:2$ compressor. A partial PPST consisting of 4-b adders is shown in Fig. 10. Thus, reduction of the number of XOR gates in the *vertical* path is done at the expense of a *horizontal* delay which is

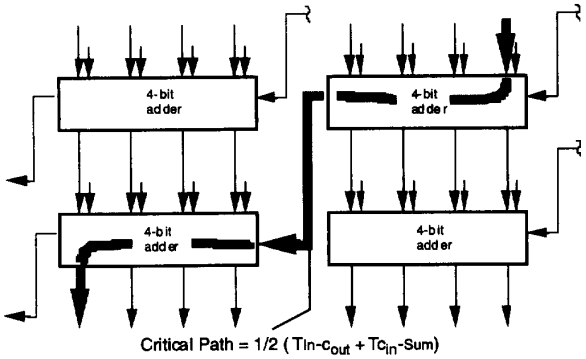


Fig. 10. Column compression tree consisting of 4-b adders (critical path involves carry propagation and one XOR delay).

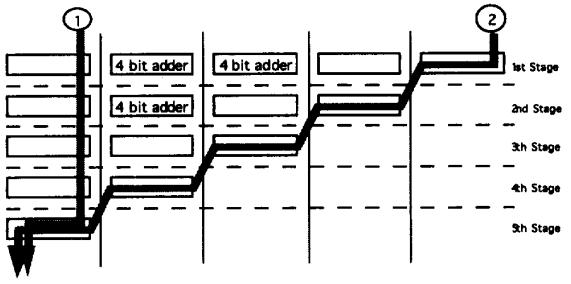


Fig. 11. Critical paths in a 24-b multiplier using 4-b adders.

assumed to be fast. An ideal case would be the use of adders which are as wide as the partial product row. In this case, the compression ratio would be two per level, and the number of XOR gate delays would also be two per level (versus three XOR gate delays for the $4:2$ compressor).

As shown in Fig. 13, the critical path for a 24-b multiplier tree using 4-b adders is calculated to be between 10 and 11 XOR delays depending on how fast the carry signal can propagate [14]. This is faster than what can be achieved with $4:2$ and $9:2$ compressors (the use of $3:2$ counters results in 14 equivalent XOR delays).

However, this case assumes the use of wide adders whose carry propagation is as fast as the sum. Since this depends on the circuit and technology and is not generally achievable, it is

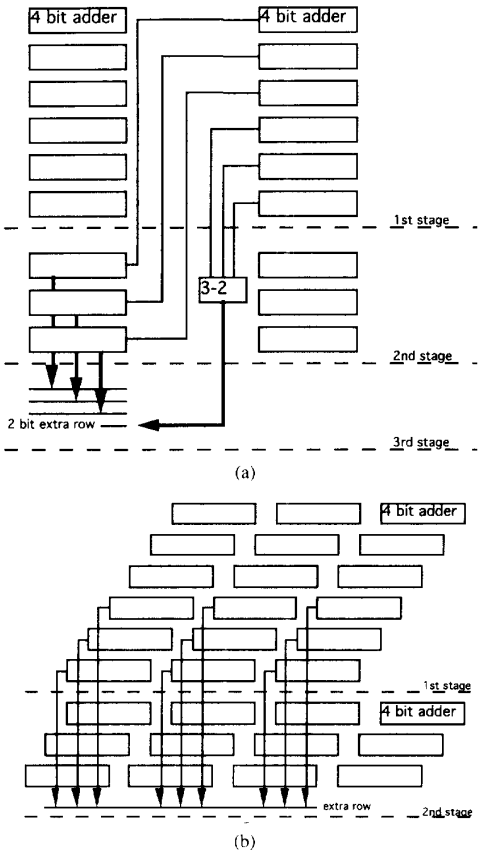


Fig. 12. Two options to absorb the extra carries.

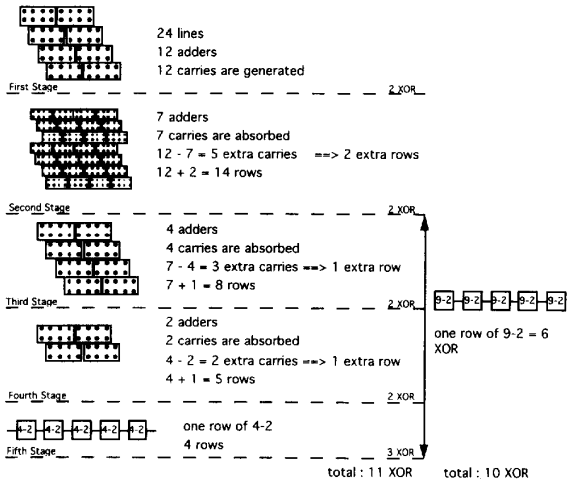


Fig. 13. 24 × 24-b integer multiplier using 4-b adders in the first stages and either 4-2 or 9-2 compressors for the last stages.

TABLE II
DELAYS IN THE 24 × 24-b MULTIPLIER TREE (T_s: DELAY OF THE SUM PER LEVEL, T_c: THE DELAY OF THE CARRY)

Path	Delay
1	$5T_s + T_c$
2	If $T_c > T_s$ then $T_s + 5T_c$ else $5T_s + T_c$

more realistic to consider shorter adders. Achieving $T_c = T_s$ is realizable in ECL technology where logic structures are built as a single transistor tree, the height of which is usually $H < 4$ [12]. Fig. 11 illustrates possible critical paths for a 24-b multiplier tree using 4-bit adders. There are several possible critical paths depending on the ratio of the speeds of the carry signal relative to the sum. The equations of these paths are given in Table II. These results assume that the depth of the tree is five levels (as in the case of the 24-b multiplier).

2) Assimilation of Extra Carries: The fact that the size of the adders K is limited leads to the use of a carry save form. Thus, it is necessary to collect carry signals after each 4 b. Fig. 12(a) depicts one method of assimilating the carries. Since the number of adders decreases from one level to the next, it is not possible to assimilate all the carries generated in the previous stage by simply connecting them to the carry-in inputs. This results in extra carries which are reduced by the help of the (3, 2) counter, thus eliminating extra rows. Although these extra carries are a major disadvantage of the adder approach, this does not greatly affect the compression ratio per level if the number of rows of partial products is large.

Another alternative method is to tile the adders in a staggered arrangement so that the carry-out signal from the K-b adder in the stage j can always be connected to a carry-in input in the next stage $j+1$. Given that the number of adders in the stage $j+1$ is less than the number of adders in the step j there will be a number of extra carry-out signals. Excess carries that cannot be absorbed as carry-in inputs form an extra row in the stage $j+1$, for every K rows of adders in the stage j. This method is shown in Fig. 12(b).

The problem of excess carries becomes critical in the last levels where the number of rows is small and the ability to assimilate carry signals is reduced. A possible option is to simply use compressors in the last levels. For example, in Fig. 13, two workable schemes using 4 : 2 or 9 : 2 compressors are proposed. The resulting number of XOR gate delays in the critical path is 11 if 4 : 2 compressors are used in the last stages. This number is further reduced to 10 with the use of the 9 : 2 compressors. However, we will show on the 24 × 24-b multiplier example how the extra carries collapse into a single row, which is always assimilated by the adders in the next level. This is repeated until the last stage, which becomes a single row of (3, 2) counters (Fig. 14).

The best approach to be taken depends on the size of the multiplier, the ratio of T_c/T_s which determines the size (K) of the adders used in the PPST and of course, technology. There is a trade-off between larger adders (which means fewer carry signals introduced) and shorter adders which are easier to realize, but generate more carry bits that need to be assimilated. The critical issue, however, is the ability to design a K-b adder such that generation of carry-out bit is as fast as sum.

3) 24 × 24-b Multiplier Example: In this example, we show how the partial product rows of a 24 × 24-b parallel

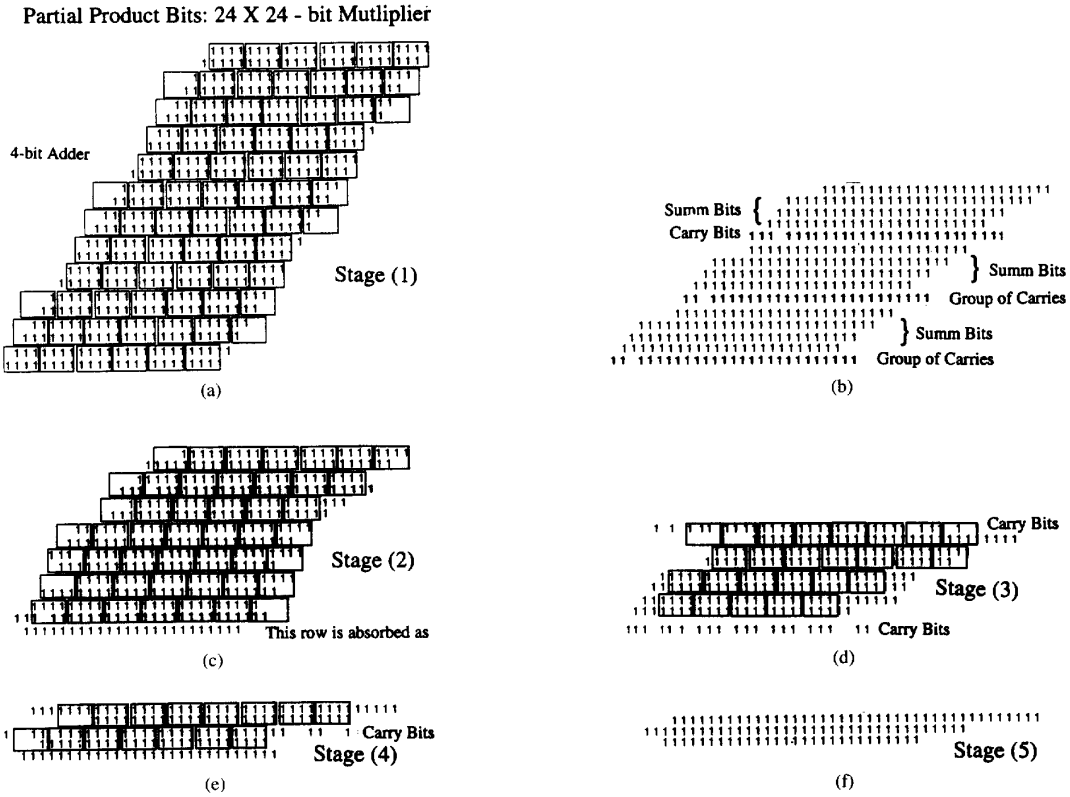


Fig. 14. PPST in a 24 × 24-b integer multiplier example.

multiplier are compressed using 4-b adders. The partial product matrix consisting of 24 rows of 24-b partial products is shown in Fig. 14(a). The 4-b adders are placed in such a way that the ones in the next row are always shifted from one bit position to the left with respect to the previous row of adders. This assures that every four rows of 4-b adders will produce only one row of carries (since the carries are skewed for one bit position with respect to each other), Fig. 14(b). The first stage (Fig. 14(a)) uses 12 adders and results in 15 rows; 12 rows are the sum bits produced by the 12 adders and three rows are the carries generated by the 12 rows of adders. Four rows of adders positioned according to Fig. 14(b) would produce one row of carries. The 15 rows of reduced partial products are further compressed by seven rows of 4-b adders in stage two. This is done by connecting 14 rows of regular inputs to 4-b adders and one row as carry inputs in the four rows of adders. Tiling of the adders allows all of the bits of this row to be absorbed (Fig. 14(c)). In the next stage (stage 3), the resulting nine rows are compressed to five rows. Of the nine rows, seven are sum bits produced by the adders in stage 3 and the rest are carry-out signals. In stage 3, we use four rows of 4-b adders, absorbing eight rows as inputs and one row as carry-in into the adders (Fig. 14(d)). The result is five rows (four sums and one row of carry-out signals) which are further compressed (in a similar manner) using two rows of 4-b adders in stage 4 (Fig. 14(e)), thus resulting in three rows at stage 5. The last three rows (in stage 5) are compressed using one row of (3, 2) counters (Fig. 14(f)).

The resulting delay consists of four stages of 4-b adders (two XOR gate delays) and a row of (3, 2) counters (two XOR gate delays) resulting in ten XOR gate delays in the critical path of a PPST of a 24 × 24-b multiplier. This is better than the PPST which uses (3, 2) counters or the one that uses 4 : 2 or even 9 : 2 compressors, yielding 14, 12 and 11 XOR gate delays in the critical path respectively. The comparison is valid only if an implementation of the 4-b adder has a worse case delay of two XOR gate delays, meaning that the carry-out signal can be produced in the same time as the sum bits.

4) Using Longer Adders: The generation of extra carries takes place in the critical part of the multiplier, i.e. in the middle of the partial product array. It may seem that the solution to the critical path problem would be to use longer adders and then to position them in the middle of the partial product array in such a way that the carry-out signals are outside of the middle range. This would requre a relatively very long adder or, a very fast carry propagation which is possible using technology such as ECL or Hitachi's double pass transistor logic (DPL) [19–20]. One such ECL implementation of a parallel multiplier had been reported by Bewick [12].

The scheme using 4-b adders was implemented using a LSI 100K ASIC library with fixed cell structure. Using that library we found that is was not possible to achieve equal delay for

carry and *sum* signals from the 4-b adder section and the delay of the carry signal was two times slower than sum. In spite of the slow carry, the speed is comparable to what was achieved using 4 : 2 and 9 : 2 compressors. Our results were obtained by simulation using LSI Logic Corporation's C-MDE simulator with LSI 100K as a target library and 1 μm CMOS technology [22].

III. Speed Improvement in the Final Adder

Finally multiplier speed is improved via optimization of the CPA to the nonuniform signal arrival profile of its inputs. It is well known that the signals applied to the inputs of the CPA arrive first at the ends of the CPA and the last ones are those in the middle of the CPA. Different shapes of the signal arrival profile originating from the multiplier tree are shown in Figs. 5, 7, and 9.

A. The Choice of Final Adder

All known schemes for fast addition were developed under the assumption that the input signals should arrive at the same time. This assumption is not realistic and we therefore are concerned about which one of the schemes for addition is most adequate as a CPA for the multiplier. It is obvious we would like to use the fastest scheme since final addition time adds directly to the critical path of the multiplier.

There are basically three regions to be considered with respect to the worst case signal arrival profile from the multiplier tree as shown in Fig. 15. The first region has a positive slope with respect to the bit position. To use an adder which adds faster than this slope would simply not make sense and would be a waste of hardware resources. The type of the adder used in Region 1 is determined by this slope. For most parallel multipliers a simple ripple-carry adder will serve the purpose. This slope is determined by the fact that the arrival of bits is incrementally delayed by a path traversing a simple one-bit adder used in bit compression. Therefore using unneccesarily powerful and complex schemes for this part of the multiplier is not justified. If the ripple-carry adder speed can not meet this slope, good choices are simple and efficient VLSI schemes such as the variable block adder (VBA) which is optimized for this particular signal arrival profile [16], [17].

From the point of the maximal delay (M), which is usually in the middle bit position or skewed a few bits toward the most significant side, the addition needs to be the fastest possible. This is because this addition time Δ_{ADD2} is a direct addendum to the multiplier delay. The choices for the adder in the Region 2 are: conditional sum adder (CSA) and carry select adder (CSLA). The carry select adder uses less hardware resources than the CSA and can be treated as its subset. The difference in speed between CSA and CSLA diminishes as the input arrival profile is changed from uniform to nonuniform [10]. Though the CSLA adder is still slower than the CSA adder, the difference in speed is offset by the relative simplicity of its implementation with respect to the CSA. This is a helpful finding given that most designers prefer CSLA over CSA. The smaller size and simpler layout of the CSLA would further affect the relative speed difference reducing the advantage of

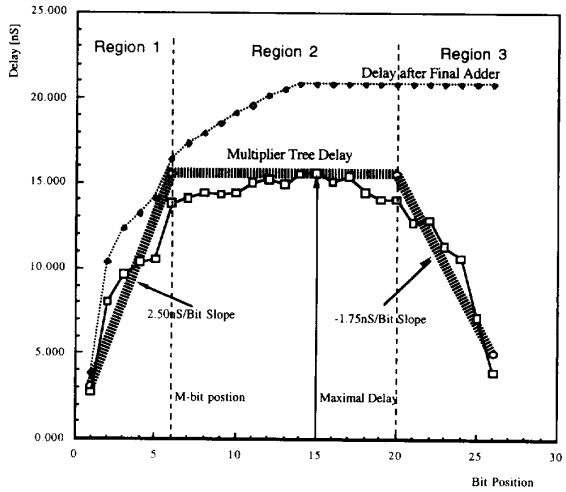


Fig. 15. Selection of the adder types in the three regions of the multiplier.

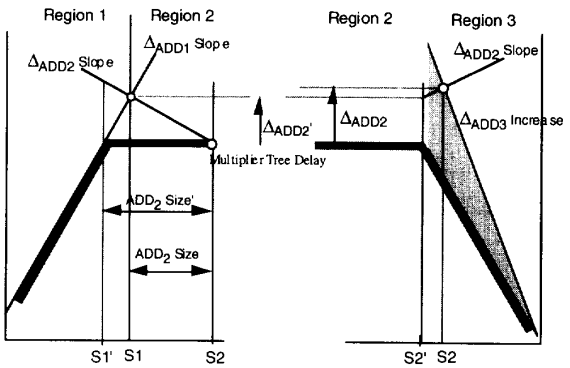


Fig. 16. Determination of bit positions S_1 and S_2 determining the size of the adders used.

CSA. Given those facts, carry select addition (CSLA) is the best choice for the Region 2 [15]. The individual adder used in the Region 2 requires the use of the fastest adder available in CSLA arrangement. A good choice is carry-lookahead adder (CLA), or a faster optimized derivative of CLA [18].

In the region of the negative slope (Region 3), the most significant bits of the input to the CPA arrive first. This favors CLA and CSLA arrangement since the addition in the most significant region can be accomplished while awaiting the arrival of carry signal from the least significant region in order to make a proper selection of the sums. The carry select adder is again the most suitable choice. However, the CSLA adds time for the selection multiplexers Δ_{MUX} which is not negligible given the adder sections are already constructed to be in the same speed range. It is also obvious that due to the steep negative slope in this region, there is substantial time left to add the bits in the most significant position before a selection process occurs. Therefore, it is possible to use a simpler scheme for implementing individual adders for the CSLA of that region. For this part the VBA [16] optimized for a particular signal arrival profile is a better choice than a

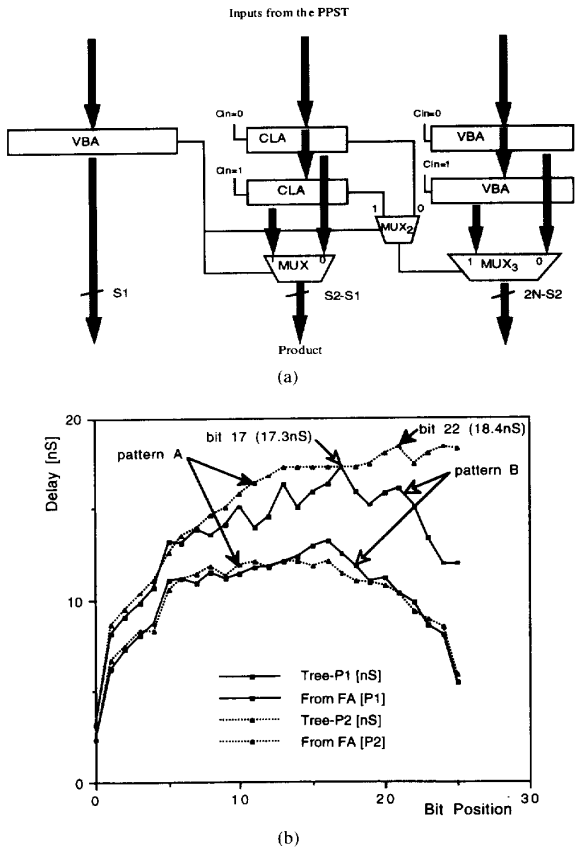


Fig. 17. (a) Structure of the final adder (b) signal arrival profile from the column compression tree and the final adder 12 × 12-b 2's complement multiplier with Booth encoding, LSI 100K [13].

CLA. Depending on the size of the multiplier, the adder in the Region 3 could consist of a CLA-VBA combination in a CSLA arrangement.

Determination of bit positions separating regions 1, 2 and 3, which determines the size of the adders used in each section of the CPA is illustrated in Fig. 16. This is an iterative technique which does not require a large number of iterations, given that the bit positions S_1 and S_2 are integers.

The total delay of the multiplier is

$$\Delta_{\mathrm{MULT}} = \Delta_{\mathrm{TREE}} + \Delta_{\mathrm{ADD2}} + \Delta_{\mathrm{MUX2}} + \Delta_{\mathrm{MUX3}}$$

The structure of the CPA used in an 12 × 12-b multiplier is shown in Fig. 17(a), and the signal arrival profile for the two different input patterns applied are shown in Fig. 17(b). Pattern B results in a faster signal arrival from the PPST than pattern A. Yet pattern B results in a longer delay than the pattern A when the CPA is included. This example illustrates the point that it is more important to *tune* the CPA into the signal arrival profile than to apply the fastest available addition scheme.

IV. Conclusion

In this paper, we have discussed ways of compressing the bits of the multiplier tree. We examined ways of designing and using different compressors and concluded that the use of full adders is warranted because the propagation delay per stage is close to two equivalent XOR gate delays (with proper design of the carry path) and the compression ratio c_r is close to 1.0 per equivalent XOR gate used in the path. Delay for various compressors in a 24-b multiplier is shown in Table III.

TABLE III
Delay for Various Compressors Applied

Compressor Applied	No. Stages	Delay of 24-bit multiplier (equiv. XOR)	No. Stages	Delay of 53-bit multiplier (equiv. XOR)
3:2	7	14	9	18
4:2	4	12	5	15
9:2	2	11	3	14
4-bit adder	5	$10+T_C$ (10)	6	$14+T_C$ (12)

As can be observed from Table III, the use of 4-b adders begins to show an advantage as the size of the multiplier increases (such as 53 b). Application of carry-propagate adders in the multiplier tree creates an extra row of carry-out signals from the adders which has to be compressed. Those bits can be absorbed at the ends of the tree because there are empty carry inputs available in the next stage of the adders applied in the PPST. The problem is critical only in the middle of the PPST. However, this extra row is eliminated by careful positioning of the adders (tiling) in a way which enables each carry produced in the previous stage to be absorbed in the next. Application of the described PPST design method results in a delay of 10 and 12 equivalent XOR delays in a PPST of an 24-b and 53-b multiplier respectively.

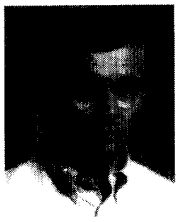
Acknowledgment

The authors would like to gratefully acknowledge the assistance of S. Liu for simulating various compressor configurations, and R. Strandberg and J. Lee for careful reading. They would particularly like to thank R. Ulicki of LSI Logic Corp., who has generously supported them with libraries and ASIC tools.

References

[1] E. E. Swartzlander, *Computer Arithmetic*, vols. 1–2, IEEE Comput. Soc. Press, 1990.
[2] K. Hwang, *Computer Arithmetic: Principles, Architecture and Design.* New York: Wiley, 1979.
[3] L. S. Wallace, "A suggestion for fast multipliers," *IEEE Trans. Comput.*, vol. EC-13, pp. 14–17, Feb. 1964.
[4] L. Dadda, "Some schemes for parallel multipliers," *Alta Frequenza*, vol. 34, Mar. 1965.
[5] A. Weinberger, "4-2 carry-save adder module," *IBM Tech. Disclosure Bulletin*, vol. 23, Jan. 1981.
[6] W. J. Stenzel, "A class of compact high speed parallel multiplication schemes," *IEEE Trans. Comput.*, vol. C-26, no. 10, pp. 948–957, Oct. 1977.
[7] M. R. Santoro, "A pipelined 64 × 64 b iterative array multiplier," in *Proc. Dig. Tech. Papers Int. Solid-State Circ. Conf.*, Feb. 1988.
[8] M. Nagamatsu *et al.*, "A 15 nS 32 × 32-bit CMOS multiplier with an improved parallel structure," presented at *Proc. 1989 Dig. Tech. Papers, IEEE Custom Integr. Circ. Conf.*
[9] J. Mori *et al.*, "A 10 nS 54 × 54 b parallel structured full array multiplier with 0.5μ CMOS technology," *IEEE J. Solid State Circ.*, vol. 26, no. 4, Apr. 1991.
[10] A. K. W. Yeung and R. K. Yu, "A self-timed multiplier with optimized final adder," Univ. California, Berkeley, Final Rep., CS 292I, Fall 1989.

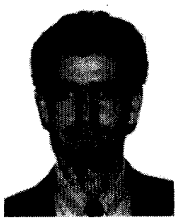
[11] P. Song and G. De Michelli, "Circuit and architecture trade-offs for high speed multiplication," *IEEE J. Solid State Circ.*, vol. 26, no. 9, Sept. 1991.
[12] G. Bewick, "Fast multiplication: Algorithms and implementation," Ph.D. dissertation, Elec. Res. Lab., Stanford Univ., Feb. 1994.
[13] V. G. Oklobdzija, T. Soulas, and D. Villeger, "An integrated multiplier for complex numbers," *J. VLSI Sign. Process.*, vol. 7, no. 3, May 1994.
[14] V. G. Oklobdzija and D. Villeger, "Multiplier design utilizing improved column compression tree and optimized final adder in CMOS technology," in *Proc. 10th Anniv. 1993 Int. Symp. VLSI Tech., Syst., Applic.*, Taipei, Taiwan, May 12–14, 1993.
[15] O. J. Bedrij, "Carry-select adder," *IRE Trans. Elec. Comput.*, June 1962.
[16] V. G. Oklobdzija and E. R. Barnes, "Some optimal schemes for ALU implementation in VLSI technology," in *Proc. 7th Symp. Comput. Arith., ARITH-7*, Urbana, IL, June 4–6, 1985.
[17] ——, "On implementing addition in VLSI technology," *IEEE J. Parallel Process., Distribut. Comput.*, no. 5, 1988.
[18] B. D. Lee and V. G. Oklobdzija, "Delay optimization of carry-lookahead adder structure," *J. VLSI Sign. Process.*, vol. 3, no. 4, Nov. 1991.
[19] M. Suzuki *et al.*, "A 1.5 nS 32 b CMOS ALU in double pass-transistor logic," in *Proc. Dig. Tech. Papers, 1993 IEEE Solid-State Circ. Conf.*, San Francisco, Feb. 24–26, 1993.
[20] N. Ohkubo *et al.*, "A 4.4-nS CMOS 54 × 54 b multiplier using pass-transistor multiplexer," in *Proc. IEEE 1994 Cust. Integr. Circ. Conf.*, San Diego, May 1–4, 1994.
[21] D. Villeger, "Fast parallel multipliers," Ecole Superieure d'Ingenieurs en Electrotechnique et Electronique, Noisy le Grand CEDEX, France, Final Rep., May 11, 1993.
[22] *1.0-Micron array-based products databook*, LSI Logic Corp., Sept. 1991.



Vojin G. Oklobdzija (S'78–M'82–SM'88) received the Dipl. Ing. (M.Sc.E.E.) degree in electronics and telecommunications from the Electrical Engineering Department, University of Belgrade, Yugoslavia, in 1971, and received the M.Sc. and Ph.D. degrees in computer science from the University of California, Los Angeles, in 1978 and 1982, respectively.

He was a Member of the Faculty of the University of Belgrade until 1976. Currently, he heads the Advanced Computer System Design Laboratory, Electrical and Computer Engineering Department, University of California, Davis. Before an early retirement in 1991, he was a Research Staff Member for eight years with the IBM T. J. Watson Research Center, Yorktown Heights, NY, making contributions to the development of RISC and super-scalar RISC architecture and processors, and he coholds one of the patents on the IBM RS/6000-"PowerPC' architecture. He has also worked on massively parallel machines and was one of the initiators of the supercomputer project at IBM. In addition, in 1988–1990, he taught courses in computer architecture, computer arithmetic, and computer design at the University of California, Berkeley, as a Visiting Faculty from IBM. His other industrial experience includes a position at the Microelectronics Center of Xerox Corporation and consulting positions at Sun Microsystems Laboratories, AT&T Bell Laboratories, and various others. He holds four USA and four European patents in the area of circuits and computer design and has published over 60 papers in the areas of circuits and technology, computer arithmetic, and computer architecture. His work on fast ALU scheme has been widely referenced and used. He has also given many invited talks in the USA, Europe, Latin America, Australia, and Japan.

Dr. Oklobdzija is on the editorial board of the *Journal of VLSI Signal Processing*.



David Villeger received the Diplome d'Ingenieur in microelectronics from Ecole Superieure d'Ingenieurs en Electrotechnique (ESIEE), Paris, France, in July 1993 after a six-month project at the University of California, Davis.

He spent time in Japan giving presentations at Yokogawa Denki Corporation and Osaka University. His current interest is in computer arithmetic and VLSI. He worked on square root and multiplication algorithms in France and in the USA, and published six papers on the subject. He is a Consultant in Computer Security and Computer Communication in the Bay Area, CA, where he worked for Sun Microsystems, and is currently consulting for Silicon Graphics Corporation.