A CMOS HIGH-SPEED ARRAY MULTIPLIER BASED ON (2,2,3) COUNTERS

BAKRI MADON
Malaysian Institute of Microelectronic Systems
7th Floor, Exchange Square
Damansara Heights
50490 Kuala Lumpur
Malaysia

ABSTRACT

An iterative array multiplier architecture based on (2,2,3) counters which is ideal for VLSI implementation is discussed. An investigation is made on the efficiency of operation speed of the counters relative to a full-adder and the viability of the multiplier architecture in CMOS technology. Circuit implementation were optimized and contrasted for speed and complexity. The comparison of the array architecture with conventional full-adder based carry-save array multipliers revealed that a significant improvement in speed can be expected with a moderate increase in hardware.

INTRODUCTION

In the pursuit of higher performance systems, the multiplication operation is recognised as one of the most vital operations in many computer and digital signal processing applications. Because of the inherent complexity of the operation, the execution speed of multiplication tends to be the dominant factor in the entire processing time. With the advent of VLSI technology, which brought about the reduced cost of fabrication per transistor on silicon, parallel array algorithms for multiplication has become increasingly important and more viable.

In a conventional carry-save array (CSA) multiplier[1] of two n bit numbers, all the partial products are generated at once and the addition operations between them are carried out by

an array of $n(n-1)$ full-adders interconnected as shown in Figure 1 for a 5 x 5-bit operation. The multiplication speed is expressed by the delay time associated with the array cells. In general, the worst-case delay D_{csa} is given by

$$D_{csa} = (2n - 2) \quad \text{full-adder delays,} \tag{1}$$

(excluding the delay due to the generation of partial product terms) since the longest carry propagate chain is through the right column and bottom row of the array. Since the scheme employs one cell for the main multiplication array, and with the regularity and simple interconnections between cells, it is ideal from a VLSI implementation point of view. Additional speed can be gained by the use of a Wallace tree[2], multiplier recoding techniques[2] and high-speed adders. However, these schemes introduce a high degree of irregularities and complexities in the array, especially for large operand word length n which makes them less attractive for VLSI implementation. Any enhancement in the speed of the CSA multiplier should be achieved, as far as possible through preservation of the homogeneous nature of the array in terms of the size of the cells, the interconnectivity and distribution of partial product terms in order to render the architecture ideal for VLSI implementation.

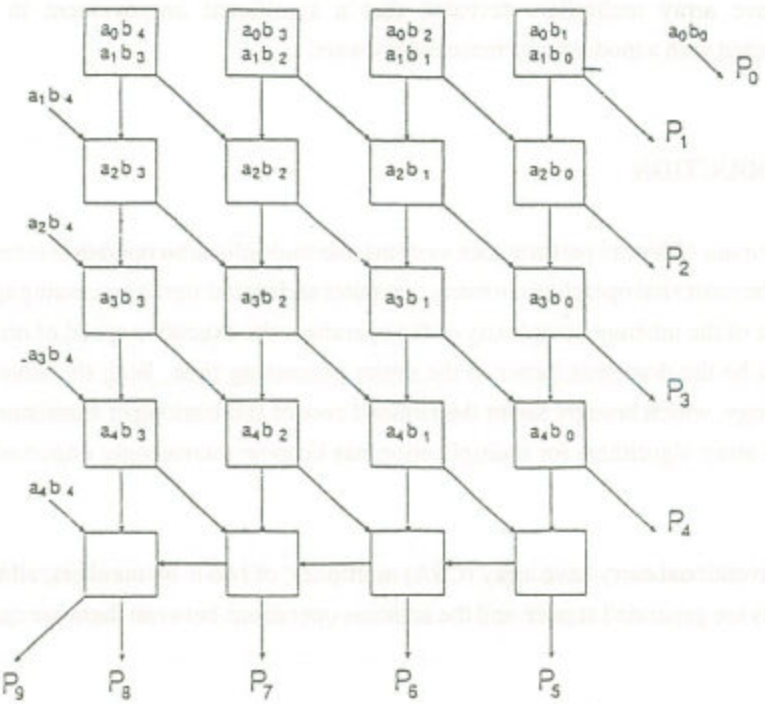

FIGURE 1. A 5 X 5-BIT CARRY-SAVE ARRAY MULTIPLIER

Significant improvements in the speed of digital array multipliers can be achieved by adding more than one partial product at a time by employing higher order parallel (p,q) counters[1-3]. This approach largely depends on a counter which has a delay and complexity comparable to that of a full-adder. Previous work has concentrated on using look-up tables[3], sequential circuits[4] and network of full-adders[5] to synthesize the counters. However, it became clear that the employment of these techniques resulted in a counter that was too slow to gain any marked improvement in the overall multiplication speed. On the other hand, the proposed architectures lacked regularity and uniformity in terms of counter size, their interconnection patterns and the distribution of partial product bits.

An iterative array multiplier based on the (5,3) counter cell[6,7] was shown to be quite close to the conventional CSA multiplier in various aspects and with a speed enhancement of nearly a factor of two, assuming the counter operated at a comparable speed to that of a full-adder. However, such a counter, with five input variables is too complicated and not expected to match a full-adder's propagation delay when implemented in practice, with the consequence that little or no speed advantage can be obtained from the (5,3) counter multiplication scheme. A (2,2,3) counter array multiplier[8,9] developed from the concept of the (5,3) counter technique revealed a promising approach to improve the multiplication speed since the (2,2,3) counter, with only four input variables can, in principle, operate nearly as fast as a full-adder. The scheme also preserves the regular interconnection structure often sought by VLSI designers.

In this paper, the (2,2,3) counter was studied, principally on the efficiency of operation speed relative to a full-adder and the viability of the array architecture in CMOS technology. Circuit techniques for realising the (2,2,3) counter were contrasted for speed against a full-adder by using SPICE circuit simulations. The architecture of a 8 x 8-bit multiplier based on the most optimum CMOS design of the (2,2,3) counter was then compared with an equivalent conventional multiplier architecture in terms of speed, hardware costs and its attractiveness for VLSI implementation. Simulation results indicated that the (2,2,3) counter could be implemented to have a delay comparable to that of a full-adder. With such a realisation of the (2,2,3) counter, the speed of the (2,2,3) array multiplier architecture would prove to be superior than that of conventional CSA multipliers.

(2,2,3) COUNTER ARRAY MULTIPLIER

The folded triangle array of partial product terms introduced by Nakamura[6,7] proved to be one of the key factors which gives the (5,3) counter multiplier its regular structure in terms of the interconnections between cells and the distribution of partial product terms.

In addition, the employment of the modified full-adder[6] and the EX-OR gate for the diagonal cells enhanced the speed of the critical path in the array by a factor of two. It can be shown that the modified full-adder and EX-OR gate composed together is essentially a (2,2,3) counter, as clearly illustrated in Figure 2. These properties are used to advantage in the development of the (2,2,3) counter array multiplier[8,9] shown in Figure 3 for an unsigned 8 x 8-bit example.

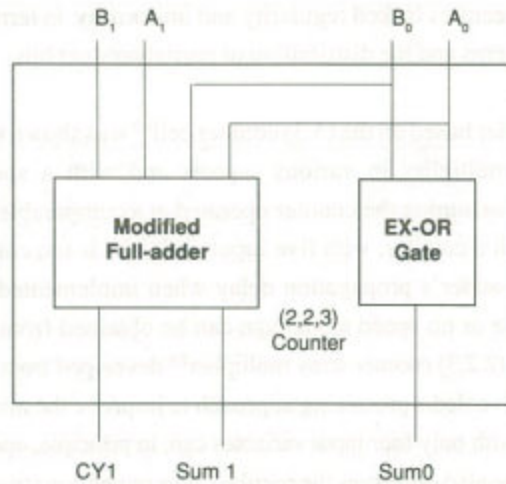

FIGURE 2. A (2,2,3) COUNTER CELL

The concept of this approach entails summing simultaneously, two consecutive partial product pairs of adjacent weights in the folded triangular matrix of partial product bits. Since this operation produces three output lines of different weights, a network of (2,2,3) counters and full-adders are interconnected to form macrocells to add up the partial product bits with the output lines from the operations of previous stages. In Figure 3, each box with partial product terms is a macrocell composed of (2,2,3) counters and full-adders whereas the diagonal cells are solely half-adders or (2,2,3) counters. There is a fourth output line (the highest carry) possible from each macrocell, depending on the number of partial product bits present as well as the number of output lines from previous stage cells (after taking their relative weighting into account). Using the coordinates shown in Figure 3, this occurs for cells $C_{i,0}$ and $C_{i,j}$ (where $i \neq j; i \neq n\text{-}1$ and $j \neq 0$). This highest carry line of such macrocell, say C_{ij} has to be computed by the cell to the right of it, that is cell $C_{i,j+1}$ in order to maintain the same number of output lines from each macrocell at subsequent stages. Furthermore, if the highest carry is instead connected to cell $C_{i+1,j+1}$ as is similarly done in the (5,3) counter multiplier, a more complicated and slower macrocell would result. The main disadvantage of connecting the highest carry this way is that, it rules out the possibility of using the same network of EX-OR gate and modified full-adder for the diagonal cells

as implemented in the (5,3) counter architecture, in the computation of the final product bits. A slightly more complicated but, still regular network of half-adders and (2,2,3) counters for the diagonal cells is needed in this architecture.

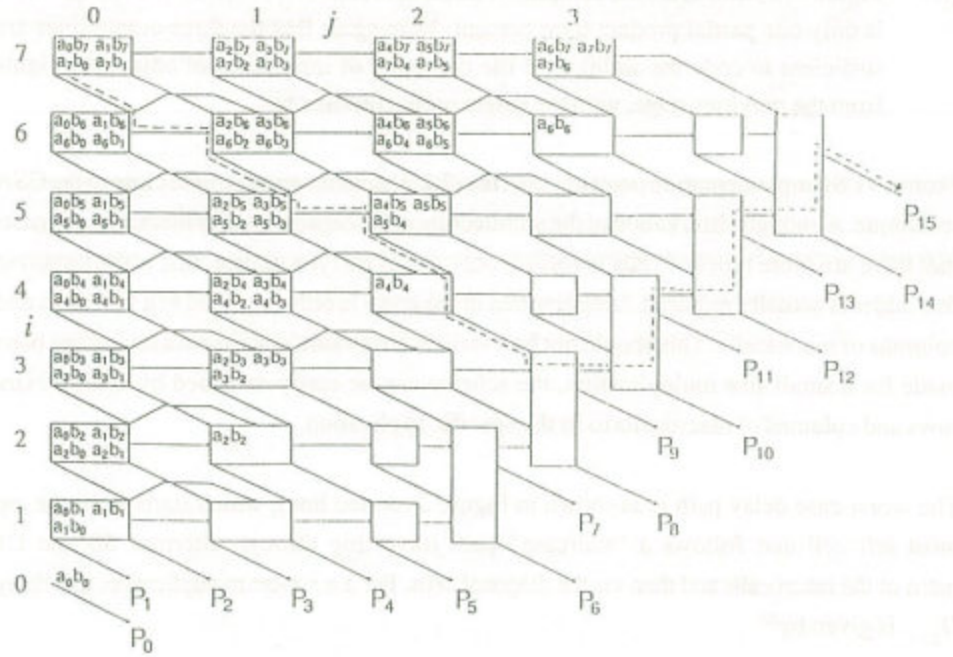

FIGURE 3. A 8 X 8-BIT (2,2,3) COUNTER ARRAY MULTIPLIER

In general, for a n x n -bit multiplication there are $(n-1)$ rows and $n/2$ columns of macrocells, which are divided into four classes as described below:

(i) For cells in the leftmost column i.e. $C_{i,0}$ (where $1 \leq i \leq (n-1)$), each cell is made up of a (2,2,3) counter *CT1*, which initially reduces the two adjacent pairs of partial product terms and another (2,2,3) counter *CT2* as depicted in Figure 4(A). One of the lower significant inputs to *CT2* is a propagating input, which comes from the S0 output of cell $C_{i+1,0}$. The propagation delay of this macrocell is one (2,2,3) counter delay for the Sl, Cl and C2 outputs whilst the S0 line does not suffer any further delay.

(ii) The composition of the topmost row macrocells $C_{n-1,j}$ (where $j > 0$) is similarly illustrated in Figure 4(B). A fourth output line C2 is not possible in this case since the full-range of the 3-bit output word is adequate to code the two pairs of partial product terms plus the bit due to the propagating signal Cl from cell $C_{n-1,j-1}$. Some minor improvements in speed and hardware savings could be achieved by replacing *CT2* in cell $C_{i,0}$ and *CT2* in cell $C_{n-1,j}$ with a (1,2,3) counter.

(iii) For macrocells $C_{i,j}$ (where $i \neq j, i \neq n-1$ and $j \neq 0$), the presence of two pairs of input lines of adjacent weights from the previous stage necessitates the use of two blocks of full-adders to add them with the summed partial product bits. This is shown in Figure 4(C).

(iv) Figure 4(D) illustrates the composition of macrocells $C_{i,1}$, where i is even and there is only one partial product term present. Note again that the three output lines are sufficient to code the addition of the two pairs of input lines of adjacent weights from the previous stage, with the single partial product bit.

From a VLSI implementation point of view, the (2,2,3) counter array is quite close to the CSA technique. Although observation of the architecture at the highest level of hierarchy suggests that there are more types of cells involved, only one extra type of gate, that is the modified full-adder is actually required. Irregularities in the array is only confined to a few rows and columns of macrocells. This should not be a serious constraint, since once a layout has been made for a small size multiplication, the scheme can be easily extended by adding extra rows and columns of macrocells to fit the specific application.

The worst-case delay path is as shown in Figure 3 (dotted line), which starts from the top most left cell and follows a "staircase" path traversing through alternate S0 and C1 paths of the macrocells and then via the diagonal cells. For a n x n-bit multiplication, this delay $D_{2,2,3}$ is given by[8,9]

$$D_{2,2,3} = (n+1) + \lfloor n/3 \rfloor \text{ cell-delays,} \quad (2)$$

by assuming that half-adders, full-adders and modified full-adders have a unit cell delay. This assumption is valid and justifiable for the following reason. A closer examination of the architecture reveals that in all the macrocells except $C_{i,\ i}$ of the (2,2,3) counter architecture, the S1 output's propagation delay, in principle would be faster than the higher carry outputs (C1 and C2) since logically, S1 is merely an EX-OR function of two variables. The implication of this is that, because the delay path through the macrocells traverses a majority of alternate logic blocks of full-adders and (2,2,3) counters, the likely slower speed of the modified full-adder component of a (2,2,3) counter would be more than compensated by the faster EX-OR gate of the (2,2,3) counter of the next stage. This makes the (2,2,3) counter multiplier a more attractive proposition than the (5,3) counter architecture since the speed requirement on the (2,2,3) counter is less stringent than for the (5,3) counter. Besides, the (2,2,3) counter, having four input variables as opposed to five for the (5,3) counter could be designed in practice with a delay comparable to that of a full-adder. This is discussed in the next section.

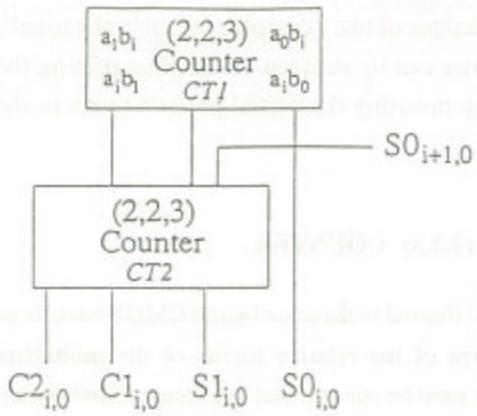

FIGURE 4(A). COMPOSITION OF MACROCELL $C_{i,0}$

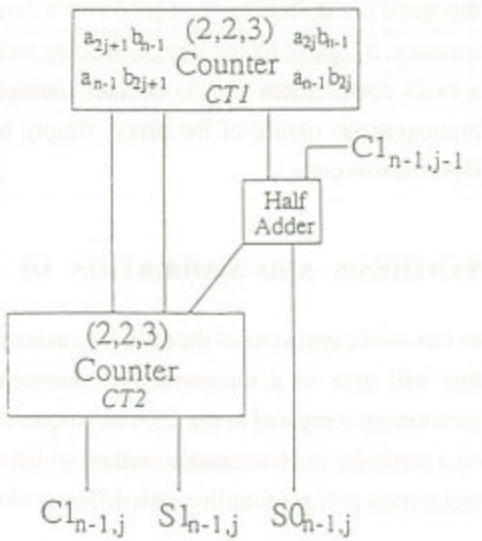

FIGURE 4(B). COMPOSITION OF MACROCELL $C_{n-1,j}$

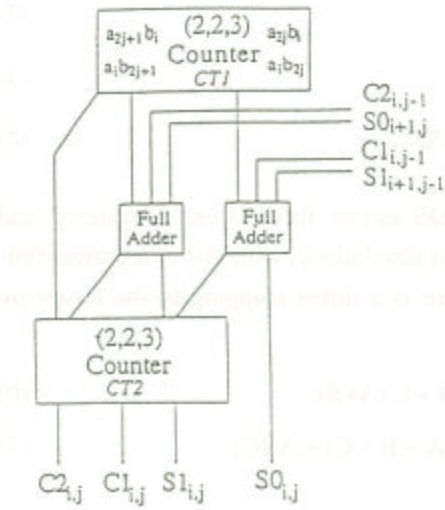

FIGURE 4(C). COMPOSITION OF MACROCELL $C_{i,j}$

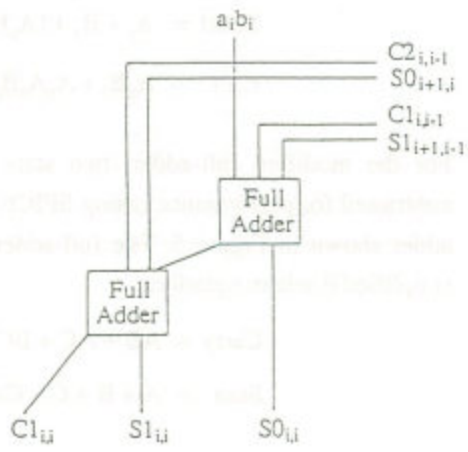

FIGURE 4(D). COMPOSITION OF MACROCELL $C_{i,i}$

For those multiplications which are based on signed binary numbers, the conversion from unsigned multiplication to signed two's complement is simple for the architecture. For instance, by applying the Baugh-Wooley technique of two's complement multiplication[10], a two's complement (2,2,3) counter multiplier can be realized, without sacrificing the homogeneous nature of the array, simply by inverting the partial product terms in the topmost row cells.

SYNTHESIS AND SIMULATION OF (2,2,3) COUNTER

In this work, synthesis of the (2,2,3) counter is limited to the use of static CMOS circuits as this will give us a representative assessment of the relative merits of the multiplier architecture compared to the CSA technique. It must be stressed that this comparison is based on a particular implementation method which may not give practical values for certain circuit techniques or logic families with different characteristics and properties.

As Figure 2 suggests, the key to high-speed implementation of the (2,2,3) multiplier is the delay of the modified full-adder since the 2-input EX-OR gate is much faster, in principle. Thus, this work only concentrates on the realisation of the modified full-adder component. The Boolean equations of the three outputs of the (2,2,3) counter are as given below:

$$\text{Sum0} = A_0 + B_0 \tag{3}$$

$$\text{Sum1} = A_1 + B_1 + (A_0B_0) \tag{4}$$

$$\text{CY1} = A_1B_1 + A_1A_0B_0 + B_1A_0B_0 \tag{5}$$

For the modified full-adder, two static CMOS circuit forms were considered and contrasted for performance (using SPICE circuit simulations) with that of a normal full-adder shown in Figure 5. The full-adder circuit is a direct mapping to the following simplified Boolean equations:

$$\text{Carry} = AB + AC + BC = AB + C\,(A+B) \tag{6}$$

$$\text{Sum} = A + B + C = \overline{\text{Carry}}\,[(A + B + C) + ABC] \tag{7}$$

It has been optimized to decrease the delay of the carry signal since it lies in the critical path during any addition operation.

In the first technique, the normal full-adder circuit is modified to accommodate the AND function of the two lower significant bits of the operands. In other words, the C input

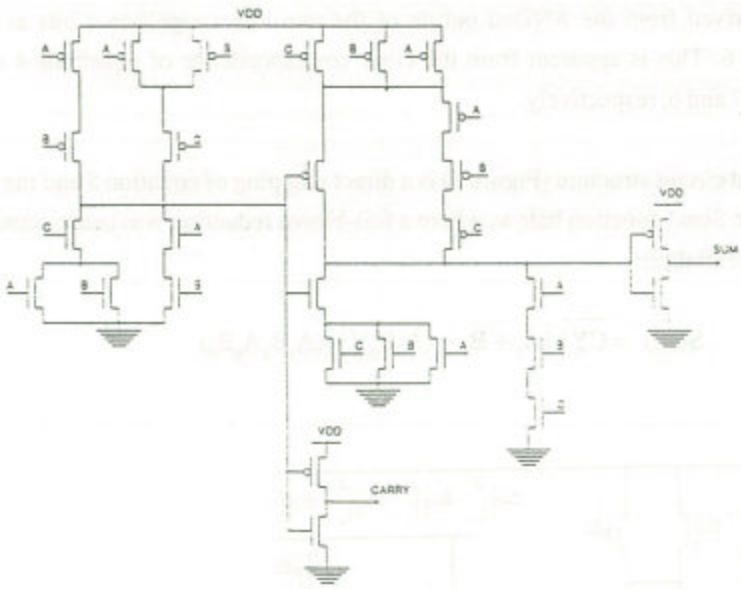

FIGURE 5. A FULL-ADDER CMOS CIRCUIT

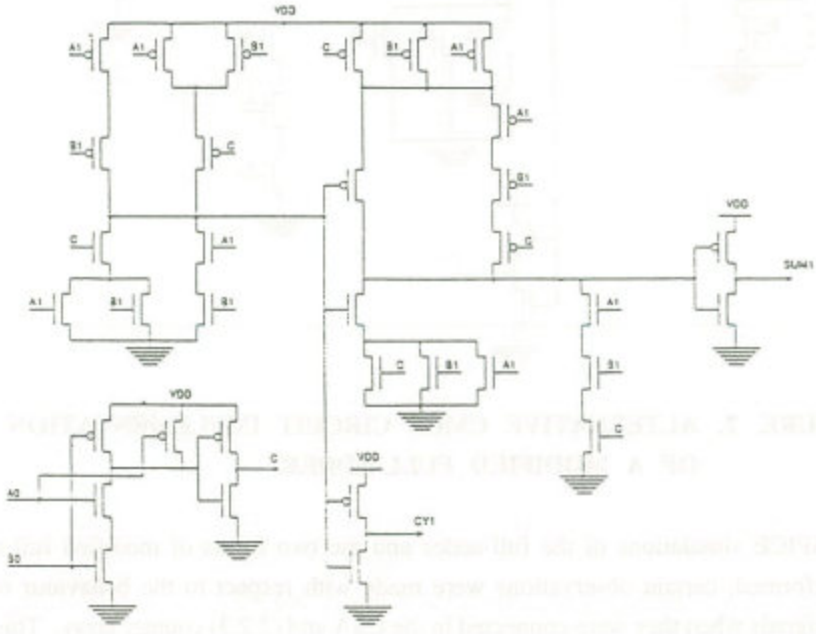

FIGURE 6. A MODIFIED FULL-ADDER CMOS CIRCUIT

is now derived from the ANDed output of the two lower significant bits as illustrated in Figure 6. This is apparent from the close correspondence of equations 4 and 5 with equations 7 and 6, respectively.

The second circuit structure (Figure 7) is a direct mapping of equation 5 and the simplified form of the Sum1 function below, where a full-blown reduction was made simultaneously on all four variables:

$$\text{Sum1} = \overline{\text{CY1}}\,[A_1 + B_1 + (A_0B_0)] + (A_1B_1A_0B_0) \tag{8}$$

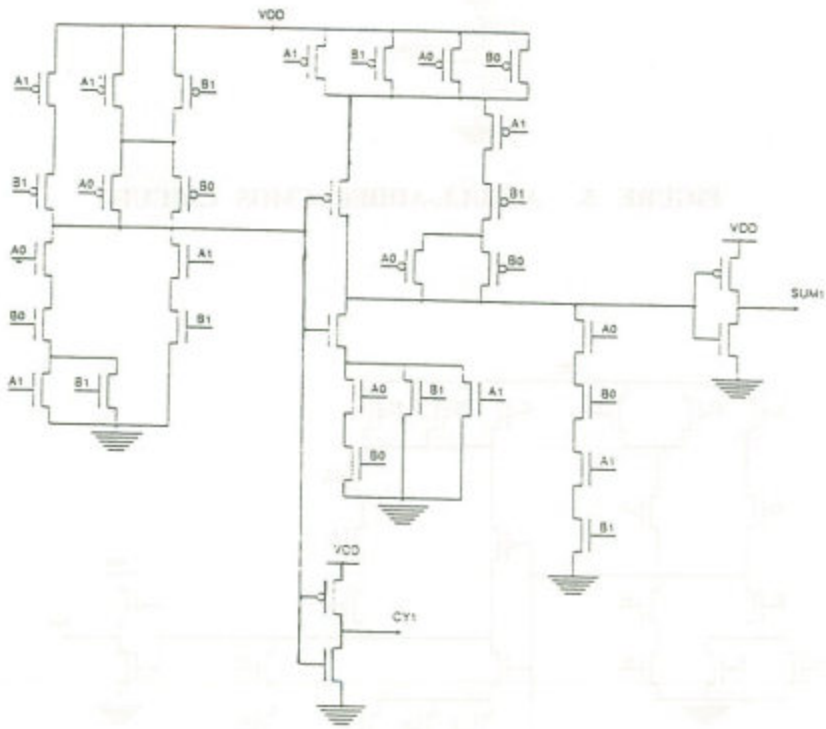

FIGURE 7. ALTERNATIVE CMOS CIRCUIT IMPLEMENTATION OF A MODIFIED FULL-ADDER

Before SPICE simulations of the full-adder and the two forms of modified full-adder were performed, certain observations were made with respect to the behaviour of the various signals when they were connected in the CSA and (2,2,3) counter array. This was necessary in order to obtain the optimum performance of the adders when interconnected in their multiplier configuration. To illustrate this, let us take a closer examination of

macrocell $C_{i,j}$. The signals C2, S0, C1 and S1 to the two full-adders are considered as propagating signals since they are ready only after the completion of operations in the connected neighbouring macrocells. We regard the three signals from counter *CT1* as steady signals because once the partial product terms stabilize, Sum0, Sum1 and CY1 are ready, right after one counter delay, and there are no propagation delays associated with them. This property of the steady signals was fully utilized when optimizing the design of the two multipliers. The input of a circuit connected to the steady signals do not have to be the one that switches the output to give a very fast response because this part only adds a constant delay for the entire multiplication time. The input that causes the output to have the fastest response should be connected to the relatively slower propagating signals.

Simulations of the normal full-adder and modified full-adder were carried out under the appropriate loading conditions in both the CSA and (2,2,3) counter array to evaluate the responses of the circuits to different input variables. It can be seen that the maximum fanout of any logic block in the (2,2,3) counter is just one more than that in the CSA multiplier. Hence, little or no degradation in the speed of the circuits can be expected when used in the (2,2,3) counter multiplier. The results of simulations of the two modified full-adders are summarized in Table 1 and Table 2 in which the Sum1 and CY1 maximum propagation delays due to different input variables switching have been normalised to the full-adder's maximum sum and carry delay, respectively. The results corresponded to the maximum possible loading conditions. Since it was observed that there was little difference in the delays for different loadings up to the maximum, the results gave a fair estimate of the relative propagation delays of the normal full-adder and the modified full-adders. In addition, for comparison purposes, it was discovered that a two input EX-OR and a two input AND gates had a normalised worst-case propagation delay of 0.5 and 0.4, respectively. Apart from being used in the macrocells, these gates were employed in the construction of half-adders found in the diagonal cells of the (2,2,3) counter multiplier.

The results showed that the implementation of the modified full-adder in Figure 6 proved to be much superior than that of Figure 7 where the worst-case speed of both functions were marginally slower than a full-adder. The poor performance of the circuit of Figure 7 can be attributed to the existence of a number of transistors connected in a long series. Some improvement can be made to Figure 6 by using more than one AND block to drive each of the C input independently. We will use the circuit implementation of Figure 6 to compare the relative merits of the (2,2,3) counter and CSA multiplier.

TABLE 1
NORMALISED PROPAGATION DELAYS OF MODIFIED FULL-ADDER CIRCUIT OF FIGURE 6

	A_0	B_0	A_1	B_1
SUM1	1.1	1.1	1.0	1.0
CY1	1.1	1.1	1.0	1.0

TABLE 2
NORMALISED PROPAGATION DELAYS OF MODIFIED FULL-ADDER CIRCUIT OF FIGURE 7

	A_0	B_0	A_1	B_1
SUM1	1.1	1.1	1.3	1.3
CY1	1.1	1.1	1.2	1.2

COMPARISON OF (2,2,3) COUNTER AND CSA MULTIPLIER

Based on the above results, the relative speeds and hardware costs can be made on the (2,2,3) counter array and CSA approach. To illustrate this, examples of a 8 x 8-bit multiplier were employed.

For a 8 x 8-bit CSA multiplier, the maximum possible delay, as given in equation 1 is 14 full-adder cell delays, excluding one AND gate delay needed for the generation of partial product terms.

In the case of a 8 x 8-bit (2,2,3) counter multiplier, from the simulation results of the last section we noted the fact that a modified full-adder had a propagation delay of 1.1 times that of a full-adder, whilst that for a 2-input EX-OR gate and AND gate were 0.5 and 0.4 times, respectively. Thus, as pointed out in Section 2, because the critical path traversed alternate blocks of 2-input EX-OR gate and modified full-adder, the relatively slower

speed of a modified full-adder was more than compensated by the much faster EX-OR gate. Using these figures to evaluate the critical path delay shown in Figure 3, this gave a worst-case speed of 10.4 full-adder delays, that is a savings of at least 3.6 full-adder delays over that of the CSA approach. This improvement in speed was shown to be consistent with the comparison determined by equations 1 and 2, and thus validated our earlier assumptions made in deriving equation 2.

To illustrate the hardware costs involved, the total transistor count of the CSA and (2,2,3) counter multiplier were calculated to be 1568 and 2420, respectively. In other words, the (2,2,3) counter multiplier represented a speed improvement of 26% at the expense of 54% increase in hardware. A much better justification between speed improvement and hardware costs could be expected especially for large operand wordlength[8,9] such as a 32 x 32-bit multiplication.

CONCLUSION

In this paper, an approach to multiplication based on an array of (2,2,3) counters and full-adders was discussed. Compared to a conventional CSA multiplier, the architecture was shown to be faster and also attractive for VLSI implementation. Since the key to the high performance of the (2,2,3) counter multiplier is the delay of the modified full-adder component of the counter, an investigation was made on the efficiency of its operation speed relative to a normal full-adder in CMOS technology. This comparison was based on the use of static CMOS circuits. SPICE circuit simulations showed that an optimized modified full-adder component of the (2,2,3) counter could be implemented to have a delay close to that of a normal full-adder. With such a realisation of the (2,2,3) counter, significant improvements in the speed of the (2,2,3) counter array multiplier over that of conventional array multipliers can be achieved with a reasonable increase in hardware.

ACKNOWLEDGEMENT

This work was mainly done in and supported by the Department of Electrical Engineering, University of Malaya, Malaysia. The invaluable assistance of Mr. Abdul Halim bin Mohd. Damiah in preparing and carrying out the SPICE simulations is gratefully appreciated.

REFERENCES

1. Habibi, A. and Wintz, P.A. Fast Multipliers. *IEEE Trans. on Computers*, 1970, C-19, 150-157.
2. Stenzel, W.J., Kubitz, W.J. and Garcia, G.H. A Compact High-speed Multiplication Scheme. *IEEE Trans. on Computers*, 1977, C-26(10), 948-957.
3. Dadda, L. Parallel Digital Multipliers. *Alta Frequenza*, 1976, 574-580.
4. Svoboda, A. Adder with Distributed Control. *IEEE Trans. on Computers*, 1970, C-19(8), 749-751.
5. Dormido, S. and Canto, M.A. Synthesis of Generalised Parallel Counters. *IEEE Trans. on Computers*, 1981, C-30(9), 699-703.
6. Nakumura, S. Algorithms for Iterative Array Multiplication. *IEEE Trans. on Computers*, 1986, C-35(8).
7 Nakamura, S. and Chu, K.Y. A Single Chip Parallel Multiplier by CMOS Technology. *IEEE Trans. on Computers*, 1988, 37(3), 274-282.
8. Madon, B. and Guy, C.G. Architectures for High-speed Multiplication. *IEEE ISCAS*, 1989, Portland, USA.
9. Madon, B. and Guy, C.G. Implementation of Parallel (p,q) Counters for High-speed Multiplication. *IEE European Conference on Circuit Theory and Design*, 1989, Brighton, UK.
10. Baugh, C.R. and Wooley, B.A. Two's Complement Parallel Array Multiplication Algorithm. *IEEE Trans. on Computers*, 1973, C-22, 1045-1047.