

# A New Design Technique for Column Compression Multipliers

Zhongde Wang, G.A. Jullien and W.C. Miller

VLSI Research Group  
University of Windsor  
Ontario, Canada N9B 3P4

## Abstract

In this paper, a new design technique for column-compression (CC) multipliers is presented. Constraints for column compression with full and half adders are analyzed and, under these constraints, considerable flexibility for implementation of the CC multiplier, including the allocation of adders, and choosing the length of the final fast adder, is exploited. Using the example of an  $8 \times 8$  bit CC multiplier, we show that architectures obtained from this new design technique are more area efficient, and have shorter interconnections than the classical Dadda CC multiplier. We finally show that our new technique is also suitable for the design of two's complement multipliers.

## Key Words :

Multipliers; Column Compression Techniques; Dadda Multiplier; Two's Complement multiplier.

## 1. Introduction

The speed of a computer or signal processor ALU depends to a large extent on the speed of the multiplier, and, over the last few decades, many high performance multiplier algorithms and architectures have been proposed [1-10]. Multiplier architectures can be classified into two categories: (1) linear parallel (LP) multipliers [1-6]; and (2) column-compression (CC) multipliers [7-13]. The carry-save adder (CSA) multiplier [1] is a typical LP multiplier. Its simple and regular structure is easily implemented in VLSI technology and it has become the most popular high performance multiplier architecture. The time delay of the LP multiplier is a linear function of  $n$ , the size of the multiplier and, for large  $n$ , the speed of the multiplier is thought to be too slow for advanced signal processors. Recently, several new architectures have been proposed for the LP multiplier [3-6] which can almost double the speed and maintain, to a certain extent, the simple structure.

The principles for the CC multiplier were established by the early work of Ofman [7], Wallace [8], and Dadda [9]. It has been shown that the delay of the CC multiplier is proportional to  $\log_{1.5} n$  [10], and the CC architecture is widely accepted as time optimal. The irregularity and complicated interconnections of the CC multiplier do not, however, readily allow efficient VLSI implementations, particularly for large  $n$ . A CMOS implementation of an  $8 \times 8$  Dadda multiplier was reported in 1988 [11], and multipliers using 4-2 compressors [12,13] and (7,3) counters [14], to improve the speed and/or regularity, were recently reported.

In the original work of Dadda [9], many different size counters were proposed to compress the columns; however, the concentration was on column compression by (3,2) counters (full adders) and (2,2) counters (half adders). Dadda showed that different schemes, including the scheme proposed by Wallace [8], require different numbers of cells (counters); the optimal scheme, which requires the least number of cells, is to compress the column size so that the height of each column follows the recursion of eqn. (1):

$$\begin{aligned} \sigma(0) &= 2 \\ \sigma(k+1) &= \left\lfloor \frac{3}{2} \sigma(k) \right\rfloor \end{aligned} \tag{1}$$

where  $\lfloor \cdot \rfloor$  represents the floor function. The series so generated is shown in (2)

$k$	0	1	2	3	4	5	6	7	8	9	...
$\sigma(k)$	2	3	4	6	9	13	19	28	42	63	...

(2)

A multiplier using such a compression scheme is normally referred to as a Dadda multiplier [10,11].

Large VLSI implementations of the Dadda multiplier have several disadvantages. The first disadvantage is the irregular structure and the associated complicated and long wiring interconnections. As an example, an  $8 \times 8$  bit Dadda multiplier [11] requires 5 cross-stage interconnections (an interconnection between non-adjacent stages), with a maximum interconnection width (measured along a single stage) of five cell widths. Most of the interconnections are not to the nearest neighbor. This irregular structure makes the VLSI layout of the Dadda multiplier difficult, and the long wiring interconnections have the potential to markedly reduce the performance.

In the compression part of the CC multiplier, adders are partitioned into stages. Except for cross-stage interconnections, the outputs of adders in one stage feed directly to the next lower stage. A natural VLSI layout of such an architecture is to arrange the cells in a pattern that renders the length of the interconnections as short as possible. For example, an  $8 \times 8$  bit Dadda multiplier requires four stages of adders for its CC part [11]. The distribution of adders is, starting from the bottom, {12,10,14,6}. If we move some adders from stage 3 to stage 4, to make the distribution of adders more even, the interconnections of those adders to stage 2, and the number of cross stage interconnections, have to be increased. This paper discusses techniques for redistributing adder cells so that local connectivity is not sacrificed.

In order to provide a figure of merit for our designs, we introduce a metric that provides an indication of the efficiency of the silicon area for CC multiplier architecture designs. Let us treat the silicon area of an adder as a unit. Since the width of the silicon area depends on the stage which contains the largest number of adders, we define the *area efficiency* of the CC part of the multiplier as:

$$\frac{N}{K \times \max(N(k))} \times 100\% \quad (3)$$

where  $N$  is the total number of the (full and half) adders for the CC part of the multiplier;  $K$  is the required number of stages, and  $N(k)$  is the number of adders in stage  $k$ . With the definition given by (3), the area efficiency for an  $8 \times 8$  bit Dadda multiplier is  $42/56=75\%$ , and for a  $12 \times 12$  bit Dadda multiplier is  $73.3\%$ .

The CC multiplier, using the 4-2 compressor or (7,3) counter as a building cell, may improve the speed and/or structural regularity [9,12-14]; however, problems associated with long wiring and area efficiency are exacerbated, as can be seen in Fig. 1. Fig 1(a) shows part of a  $9 \times 9$  bit Dadda multiplier using adders, Fig. 1(b) part of a  $16 \times 16$  bit 4-2 compressor multiplier, and Fig. 1(c) part of a  $32 \times 32$  bit multiplier using (7,3) counters. The number marked on each cell in Fig.1 indicates the binary weight of that cell. The area efficiencies of the adder, 4-2 compressor and (7,3) counter multipliers are 58%, 58% and 53%, respectively. Both the Dadda and (7,3) counter multipliers have cross-stage interconnections, but they do not have interconnections within one stage; on the other hand, the 4-2 compressor CC multiplier avoids cross-stage interconnections, but at the expense of long-wiring interconnections within each stage. The longest interconnection for the three cases are 2, 4, and 7 cell widths, respectively. Since a cell of either 4-2 compressor or (7,3) counter is wider than the adder cell, the actual maximum length of wiring for these two multiplier types is longer than the width ratios suggest. From Fig. 1 we may conclude that if the length of wiring and area efficiency are prominent cost function elements, then the Dadda multiplier is the most efficient of the three.

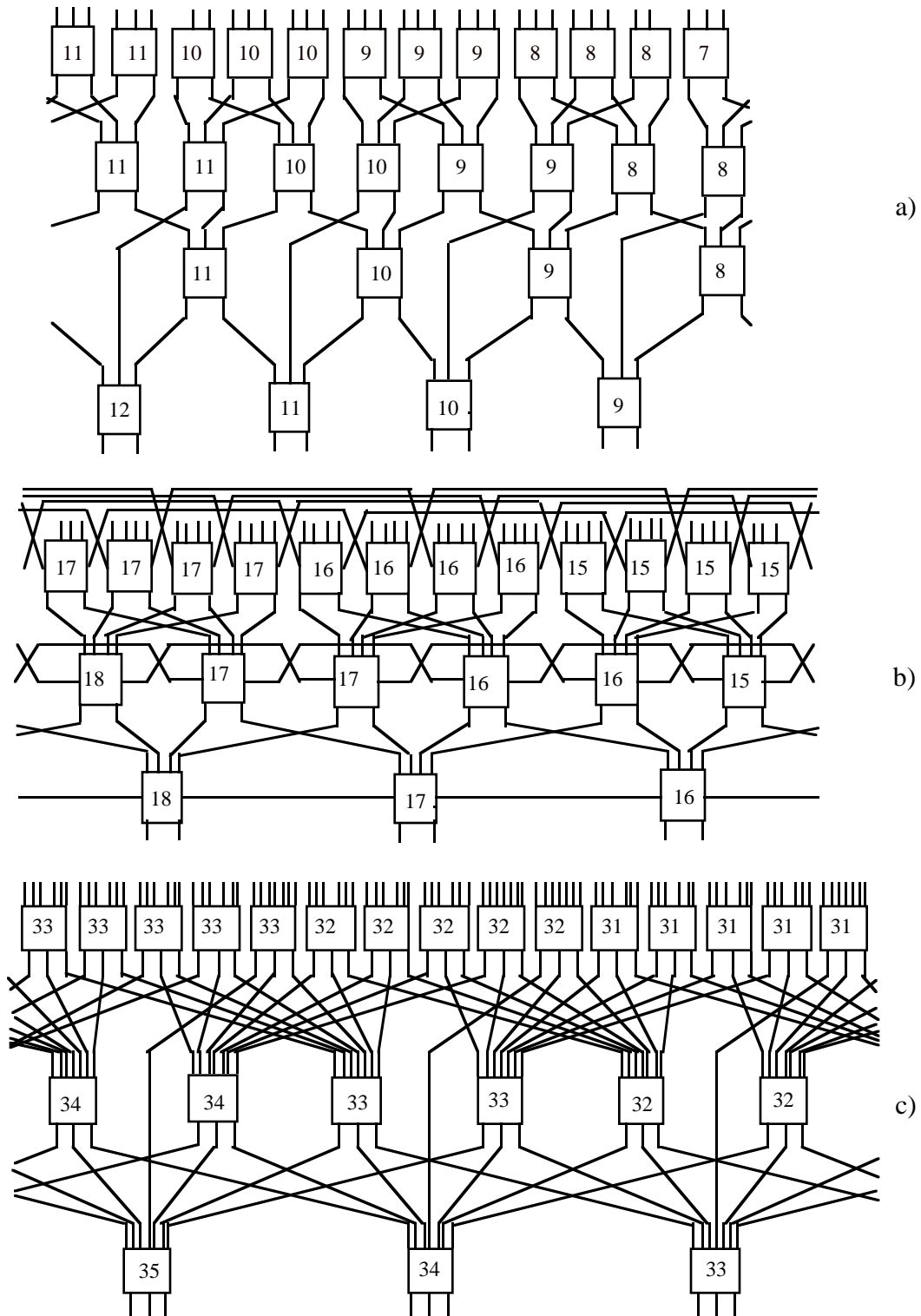


Fig.1 Typical structure and interconnections for:  
 (a) Dadda multiplier; (b) 4-2 compressor multiplier; and (c) (7,3) counter multiplier

In this paper we will show that a CC multiplier, with equal size multiplicand and multiplier and built only with adder cells, requires a minimum number of cells, and that there is considerable flexibility in allocating this number of cells to different stages. Our strategy is to maximize the area efficiency by allocating adders to each stage as evenly as possible, to eliminate as many cross-stage interconnections as possible, and to reduce the maximum length of in-stage interconnections. Our approach builds upon the pioneering work of Wallace [8] and Dadda [9].

The paper is organized as follows: we first determine the lower bounds on the number of adders required by a CC multiplier. Then we discuss the constraints governing the distribution of adders to the different stages. We then propose a procedure that attempts to maximize area efficiency while reducing the number of cross-stage interconnections. For short length CC multipliers we show that it is advantageous to shorten the length of the final adder, and we provide examples of this procedure. Finally, we consider the implementation of two's complement multipliers using our new technique.

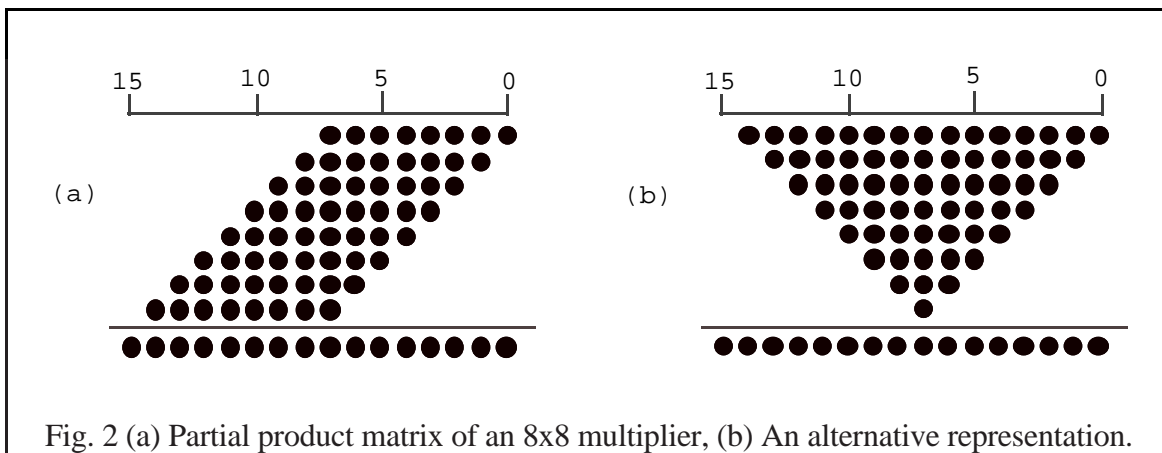
## 2. A Lower Bound on the Total Number of Adders

Dadda showed that different schemes for column compression require different numbers of adders [9]. This section determines the lowest bound of that number.

For an  $n \times n$  bit multiplier, the partial product of two  $n$  bit numbers,  $a_i b_j$   $\forall i, j \in \{0, 1, \dots, n-1\}$ , forms a matrix of  $n$  rows (an example for  $n = 8$  is shown in Fig. 2a). Each row contains  $n$  elements with a shift of one element to the left from the preceding row. Thus the matrix contains  $n$  rows and  $2n - 1$  columns and can be alternatively represented by Fig. 2b. We assign an ascending integral index, starting from 0, to columns from the right to the left. Thus the  $j$ th column indicates that the elements in that column possess a weight of  $2^j$ . All existing multipliers deal with methods to sum up the partial product matrix to obtain the final result; that is, a matrix with just one row of  $2n$  elements. Let  $p(j)$  be the number of

partial products in the  $j$ th column. From Fig. 2 it is easy to determine that:

$$p(j) = p(2n - 2 - j) = j + 1; 0 \leq j \leq n - 1 \quad (4)$$



Our purpose is to reduce the size of each column to one by using full or half adders. A full adder in column  $j$  can absorb three bits in that column and a half adder absorbs two. For both adders a sum bit goes to column  $j$  and a carry bit to column  $j + 1$ . A full adder in column  $j$  will reduce the column size by two and a half adder by one, and both adders increase the size of column  $j + 1$  by one. Clearly, full adders have priority over half adders in column reduction.

Let  $e(j)$  be the expanded size (number of entries) of column  $j$ , which includes partial products in column  $j$  and the carries from column  $(j - 1)$ . Let  $q_0(j)$  be the number of the necessary adders to reduce the expanded size,  $e(j)$ , to one.  $[e(j) - 1]/2$  full adders are needed to reduce column size  $e(j)$  to one if  $e(j)$  is odd;  $e(j)/2 - 1$  full adders and one half adder are needed if  $e(j)$  is even. The carries in column  $j$  are produced by adders in column  $j - 1$ , and their numbers are exactly the same. Thus, we can write the following recursive equation

$$e(j) = p(j) + q_0(j - 1) \quad (5)$$

where

$$q_0(j) = \left\lfloor \frac{e(j)}{2} \right\rfloor \quad (6)$$

Column 0 contains one partial product,  $a_0b_0$ , and therefore no adders are required for that column. Thus,  $q_0(0) = 0$ . With this initial condition, it is easy to obtain:

$$q_0(j) = q_0(2n-1-j) = j; \quad 1 \leq j \leq n-1 \quad (7)$$

The total (necessary and sufficient) number of full and half adders to reduce the partial product matrix to a matrix with just one row is given by

$$\sum_{j=1}^{2n-2} q_0(j) = 2[1 + 2 + \dots + (n-1)] = n(n-1) \quad (8)$$

Among these  $n(n-1)$  adders,  $n$  of them are half adders because there are  $n$  columns with an even number of entries.

Fig. 3 shows the expansion of column size by carries and the distribution of adders in different columns for the example of an  $8 \times 8$  bit multiplier. It is interesting to note that the number of full and half adders required by a CSA multiplier [1,2] is exactly given by eqn. (8). The CC multiplier uses a fast adder (for example, a carry-propagate adder) for the last step in order to change a two row matrix to a single row. The final fast adder is of length  $2n-2$ , with cells ranging from column 1 to  $2n-1$ . The cell at column 1 is a half adder, the other cells are full adders. The number of adders required by the CC part of an  $n \times n$  bit CC multiplier is thus given by:

$$N = n(n-1) - (2n-2) = (n-1)(n-2) \quad (9)$$

Among them,  $n-1$  are half adders, distributed from column 2 to  $n$ . For example, for a  $12 \times 12$  bit CC multiplier, the required number of adders is  $N = 11 \times 10 = 110$ . 11 of them are

half adders<sup>1</sup>. Representing the required number of adders in column  $j$  for the CC part by  $q(j)$ , clearly we have  $q(j) = q_0(j) - 1$ . Thus:

$$q(j) = q(2n - 1 - j) = j - 1; \quad 2 \leq j \leq n - 1 \quad (10)$$

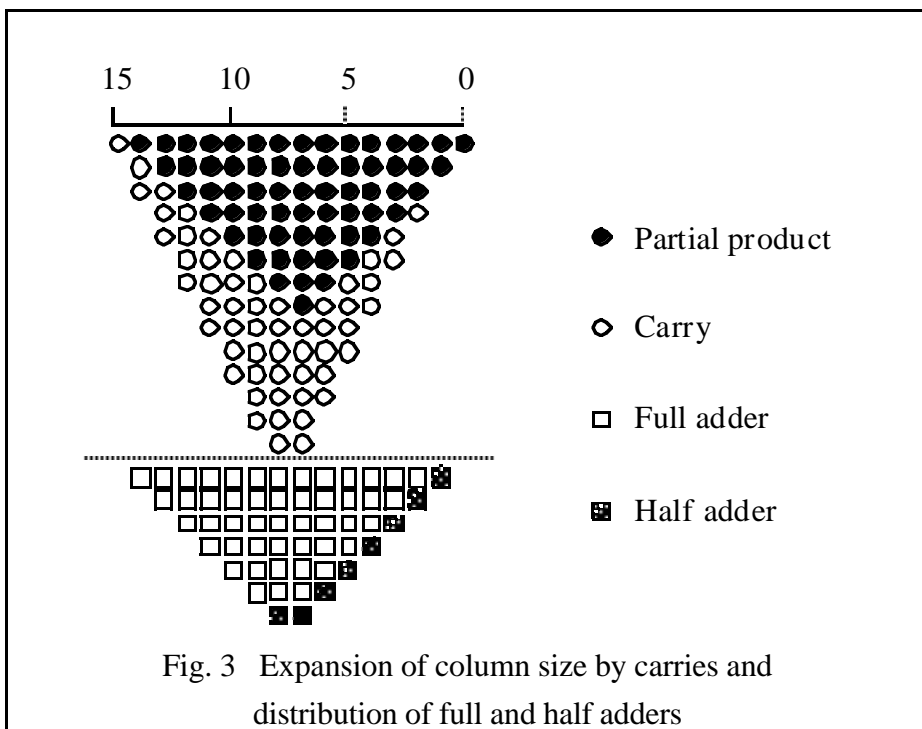


Fig. 3 Expansion of column size by carries and distribution of full and half adders

### 3. Constraints For Adder Allocation

As will be shown later, the restriction on column size in each stage [9] (according to the series in eqn. (2)) may be violated when we allocate adders to different stages. However, eqn. (2) is valid in determining the number of the required stages for the unsigned CC multiplier implemented by full and half adders. If the size of the multiplier,  $n$ , satisfies:

$$\sigma(K - 1) < n \leq \sigma(K) \quad (11)$$

---

<sup>1</sup> We note that there is an error in counting the number of half adders in [9]. The number of half adders for the scheme shown by Fig. 8 of [9] should be 10, instead of 14.

then the CC part of the CC multiplier requires  $K$  stages.

After  $N$  and  $K$  are determined, there are many ways to allocate  $N$  adders to  $K$  stages. The distribution of  $N$  adders to the  $K$  stages will strongly influence the area-efficiency and the maximum length of interconnect. Since there are many distributions which yield the same result for the same number of cells, to find a distribution which provides the highest area-efficiency and yet minimizes the length of in-stage inter-connects and number of cross-stage interconnects is of great interest. Suppose  $N(k)$  adders are assigned to the  $k$ th stage, with  $q_k(j)$  adders being assigned to column  $j$ . Obviously:

$$\sum_{j=1}^{2n-2} q_k(j) = N(k); \quad k = 1, 2, \dots, K \quad (12)$$

In each column, we assign exactly the number required by that column. Thus

$$\sum_{k=1}^K q_k(j) = q(j); \quad j = 2, 3, \dots, 2n - 2 \quad (13)$$

Each adder at column  $j$  and  $j - 1$  in stage  $k$  produces an output in column  $j$ . The number of the outputs at column  $j$  produced by adders in stage  $k$ , given by  $q_k(j) + q_k(j - 1)$ , should not exceed the number of slots (bit positions which can accept inputs) at column  $j$  provided by the lower stages, which is given by:

$$\begin{aligned} s_k(j) &= 2 + 3 \sum_{i=1}^{k-1} q_i(j) - \sum_{i=1}^{k-1} [q_i(j) + q_i(j-1)] \\ &= 2 + \sum_{i=1}^{k-1} [2q_i(j) - q_i(j-1)] \end{aligned} \quad (14)$$

The first term on the right hand side is provided by the fast adder, the second term is produced by all adders in the lower stages of column  $j$ ; the last term is the number of slots which have been occupied by the adders already allocated at column  $j$ , and the carries produced by adders at column  $j - 1$ , in the lower stages. Thus, we arrive at the constraint:

$$q_k(j) + q_k(j-1) \leq 2 + \sum_{i=1}^{k-1} [2q_i(j) - q_i(j-1)] \quad (15)$$

The purpose of allocating adders in each stage is to provide slots to accept inputs; this is providing that the total number of adders required by that column have not been used up by the lower stages. Therefore, except for those columns where no more adders are needed, the number of slots provided by stage  $k$  in column  $j$  should not be less than the number of slots provided by stage  $k-1$  in the same column; or  $s_{k-1}(j) \leq s_k(j)$ . Using eqn. (14) we find:

$$\begin{cases} q_k(j) \leq 2q_k(j+1) & j < J_k \\ q_k(j) = 0 & j = J_k + 1 \end{cases} \quad (16)$$

If  $q_k(j) = 1$ , from constraint (16) we know that we have to allocate at least one adder to each column up to column  $J_k$ , the highest index the adders to be allocated possess.

Under the constraints (12) (13), (15) and (16), together with condition (10), there are many schemes available to allocate adders to different stages using the minimum number of adders; Dadda's architecture is one such scheme. In considering the area efficiency, the best choice is to allocate adders to each stage as evenly as possible to maximize the area efficiency. There is, however, an upper bound for the number of adders in each stage, and this is derived in the following section.

## 4. An Upper Bound on Adders in Each Stage

The number of adders in a specific stage reaches its maximum if no remaining adders can be put in any column. The maximum number of adders in stage  $k$ ,  $N_{\max}(k)$ , can be derived from the constraints in section 3. We start from stage 1.

The final fast adder provides two slots to each column ranging from 2 to  $2n-2$ , and three slots, including the carry-in, for column 1. We have already shown that the adders for the CC

part start from column 2. Constraint (15) tells us that if we allocate an adder to column 2, we have to allocate adders to every other column. The only choice, then, is to allocate one adder to each column. In this case, the slots on columns 3 to  $2n - 3$  of the final fast adder will be fully occupied, and thus the number of adders in stage 1 reaches the maximum,  $N_{\max}(1) = 2n - 4$ .

The adders in stage 1 provide three slots for columns ranging from 2 to  $2n - 2$ ; therefore, we can allocate three adders to each pair of two adjacent columns. However, there is only one adder remaining not allocated on column 3 and  $2n - 3$ , and zero adders remain in columns 2 and  $2n - 2$ . Therefore, we can only allocate two adders to every other column ranging from column 4 to  $2n - 4$ . For the other columns we can allocate one adder to each column only. In this way the number of adders at stage 2 reaches the maximum,  $N_{\max}(2) = 3n - 10$ .

The number of adders that we can allocate to stage 3 relates to the number of adders we have allocated to stage 2. If we have allocated  $N_{\max}(2)$  adders to stage 2, the number of slots provided by the adders on columns of stage 2, starting from column 5 to  $2n - 5$ , will alternatively be 3 and 6. This means that we can allocate no more than three adders to every two adjacent columns. Every other column will contain three more slots which can not be occupied by any adders. Therefore, if the number of adders in the second stage reaches  $N_{\max}(2)$ , the number of adders on the third stage can not reach  $N_{\max}(3)$ . However, if we allocate only one adder to each column on stage two, starting from column 3 to  $2n - 3$ , then each column in that range will provide 4 slots to stage 3. Three of them are provided by the adder on stage 2. The last one is provided by the adder on stage 1. In this way we can allocate two adders to each column starting from column 5 to  $2n - 5$ . There are no empty slots provided by the previous stages in that column range. Thus, the number of adders on the third stage reaches its maximum  $N_{\max}(3) = 4n - 18$ , while the number of adders on the second stage reduces to  $2n - 6$ . It is interesting to note that  $\max\left\{\sum_{j=2}^3 N(j)\right\} = 6n - 24$  adders can be

arbitrarily distributed to these two stages, with the constraint that the number of adders on each

stage can not exceed  $N_{\max}(k)$ . This provides us with the flexibility to adjust the multiplier footprint for maximum area efficiency.

In the same manner, we can determine  $N_{\max}(k)$  for the other stages. Table 1 shows the results. For comparison, Table 1 also shows the number of adders for each stage of Dadda's scheme. The fourth column,  $\hat{N}(k) = \sum_{i=1}^k N(i)$ , is the number of adders contained in the first  $k$  stages that allows  $N_{\max}(k+1)$  adders to be allocated to stage  $k+1$ . In the fifth column of Table 1, we show the condition for calculating numbers of adders in columns 2 and 3 of the table.

Stage	$N_{\max}(k)$	Dadda's Scheme	$\hat{N}(k)$	
1	$2(n-2)$	$2(n-2)$	$2n-4$	$n > 4$
2	$3n-10$	$2(n-3)$	$5n-14$	$n > 6$
3	$2(2n-9)$	$2(2n-9)$	$8n-28$	$n > 9$
4	$6(n-7)$	$6(n-7)$	$14n-70$	$n > 13$
5	$9n-93$	$4(2n-21)$	$23n-163$	$n > 19$
6	$13n-205$	$12(n-16)$	$35n-357$	$n > 28$
7	$19n-433$	$18(n-23)$	$53n-779$	$n > 42$

Table 1 The Maximum Number of Adders for Each Stage

## 5. Approach I: A Procedure for Allocating Adders

In order to obtain high area efficiency, we try to allocate approximately the same number of adders to each stage, subject to the constraints and bounds derived in the previous two sections. We introduce the following algorithm.

- 1 Calculate the average number of adders for each stage,  $\overline{N}_0 = \left\lceil \frac{N}{K} \right\rceil$ .
- 2 If every stage can accommodate  $\overline{N}_0$  adders, then we allocate at most  $\overline{N}_0$  adders to each stage, with all constraints being satisfied, and with all  $N$  adders allocated

to the  $K$  stages. The algorithm terminates at this point.

- 3 If 2 does not apply and, say,  $k$  lower stages can not contain  $N$  adders, we fill those stages with as many adders as they can contain, and calculate the average number of adders for the remaining  $K - k$  stages,  $\bar{N}_k = \left\lceil \frac{N - \hat{N}(k)}{K - k} \right\rceil$ , where  $\hat{N}(k)$  is given by the fourth column of Table 1; we assume  $\hat{N}(0) = 0$ .
- 4 Check if each of the remaining  $K - k$  stages can accommodate the number of adders determined from step 3. If true, we allocate at most  $\bar{N}_k$  adders to each of the  $K - k$  stages, under the condition that all constraints are satisfied and that all  $N$  adders have to be allocated to  $K$  stages.
- 5 If at least one stage of the remaining  $K - k$  stages cannot accommodate  $\bar{N}_k$  adders, we go to step 3 and repeat steps 3-5, until an appropriate  $\bar{N}_k$  is obtained.

Since we allocate to each stage either the maximum number of adders it can contain, or a number of adders which do not exceed the appropriate average number, the above algorithm is guaranteed to maximize the area efficiency.

### 5.1 An 8x8 Multiplier Example

Let us use an  $8 \times 8$  multiplier as an example. From the series (2) and equation (9),  $K = 4$  and  $N = 42$ , and the average number of adders for each stage:  $\bar{N}_0 = \left\lceil \frac{N}{K} \right\rceil = 11$ . Since  $N_{\max}(1) = 12 > \bar{N}_0$ , we thus allocate at most 11 adders to each stage.

Next, we distribute adders of the CC part to different stages; this is a heuristic procedure. For our case, let the distribution be  $\{11,11,11,9\}$  for stages 1 through 4. Although this is a somewhat arbitrary distribution, we find that there are not many other choices. For example, it



Next, we construct a table with  $K$  rows and  $\overline{N}_0$  columns to represent the layout pattern of the CC part of the multiplier; each block represents a unit area of silicon which can physically accommodate an adder, and the blocks contain the binary weight of the adders (Fig. 4).

<b>4</b>		10	9	8	8	7	7	6	6	5	
<b>3</b>	11	10	9	9	8	8	7	7	6	5	4
<b>2</b>	12	11	10	9	8	7	6	5	4	3	2
<b>1</b>	13	12	11	10	9	8	7	6	5	4	3
	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>

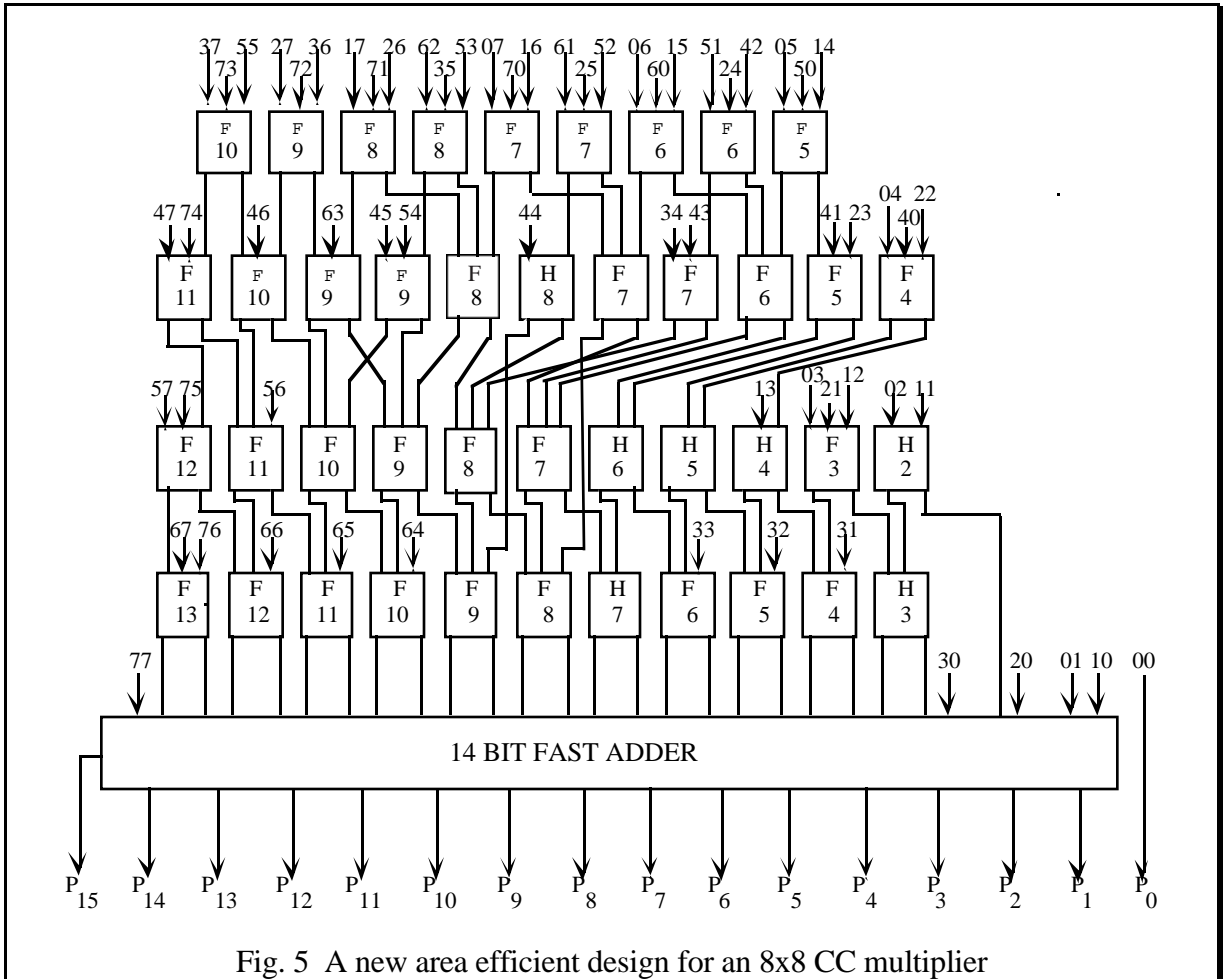
Fig. 4. Table layout of the CC part of an  $8 \times 8$  bit CC multiplier

Since each of the stages 1, 2, and 3 contain 11 adders, the assignment of adders for these three stages, as shown by Fig. 4, is completely determined. Stage 4 contains 9 adders, and the assignment shown by Fig. 4 for this stage yields the shortest interconnection.

In order to measure the length of interconnections, we refer to each adder position by its table coordinate; for instance, adder (11,2) has weight 2. We use the cell width as the length unit. We assume the layout of an adder is a square with inputs into the top and outputs out of the bottom. The connection of two adjacent adders (e. g. adder (3,3), weighted 9, to adder (3,2), weighted 10) will be counted as a length 0 connection. The length of connection between adder (i,j) and (k,l) is thus given by  $|i - k| + |j - l| - 1$  cell widths. The longest interconnection is 3 cell widths from the carry of (11,3) to input of (8,2).

Fig. 5 shows the architecture of this multiplier, where each adder is represented by a rectangular block. The adder weight is indicated by the number on the block, and full and half adders are indicated by letters F and H, respectively. The tuple  $ij$  on each input arrow represents the partial product  $a_i b_j$ . The area efficiency of the CC part for this architecture is  $42/44=95.5\%$ , which is the maximum that we can reach. There are 3 cross-stage

interconnections.

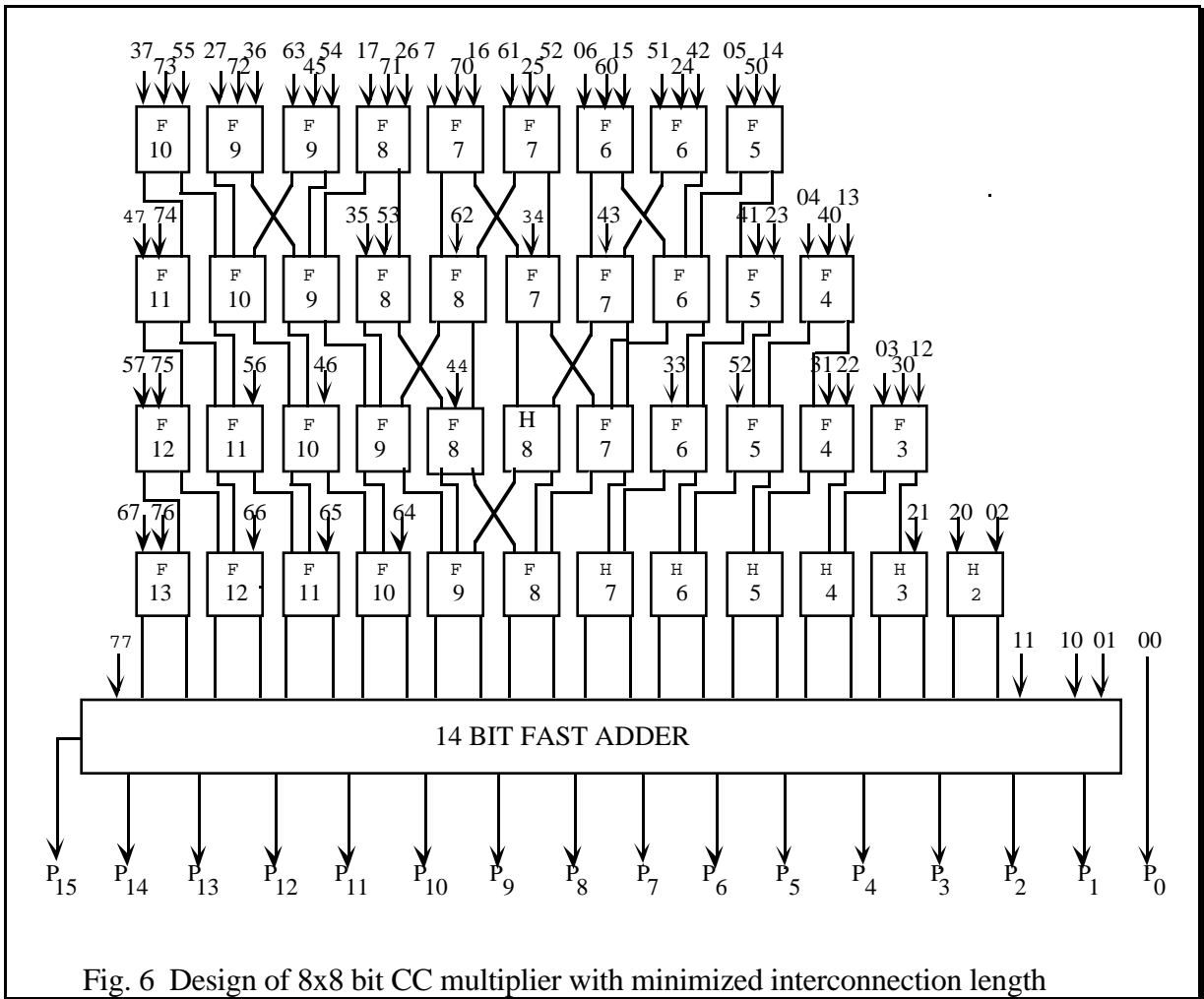


We can now compare our design to the architecture of the  $8 \times 8$  bit Dadda multiplier, given in Fig. 3 of reference [11]<sup>4</sup>, where there are 5 cross-stage interconnections with a maximum interconnection distance of 5 cell widths; by our measure the area efficiency is  $42/56=75\%$ . The architecture given in Fig. 5 is therefore more efficient than the Dadda multiplier of reference [11], both in terms of the area efficiency measure and in maximum length of interconnection. From many designs that we have tried, the architecture given by Fig. 5 has the highest area efficiency, based on our cost measures.

<sup>4</sup> We have found several errors in this figure.

5.1.1 An improved architecture

We can trade area efficiency for maximum interconnection length. Fig. 6 shows an improved architecture in this regard. The area efficiency of this architecture is  $42/48 = 87.5\%$ , 7 percent lower than the architecture given by Fig. 5; however, all cross-stage interconnections have been eliminated, and all interconnections are either to the nearest or to the next nearest neighbor. Thus, the maximum length of interconnection reduces from 3 to 1 cell widths. We feel that these advantages are well worth the 7% reduction in area efficiency.



## 6. Approach II: A New Design Technique for Short Length Multipliers

For either Dadda's scheme or the scheme given in the last section, the length of the final fast adder is  $2(n-1)$ , which is always larger than the maximum number of adders for the first stage of the CC part,  $2(n-2)$ . We can shorten the length of the final fast adder to raise the area efficiency, since each stage can reduce the length of the final adder by one bit, with a total reduction of, at most,  $K$  bits. We will refer to the extreme case, i.e., using a length  $2n - 2 - K$  adder for the last adder, as *Approach II*. The maximum number of adders and  $N_m(k)$  for approach 2 are listed in Table 4. The number of adders that the CC part contains is  $N = (n-1)(n-2) + K$  in this case. The allocation procedure is the same as described in the previous section. In some cases, it may be more advantageous to reduce the length of the final adder by a number less than  $K$ ; examples of these cases will be shown later.

As shown in Table 4, if  $K \geq 2$ , the maximum number of adders for the first two stages of the CC part of the multiplier for Approach II is less than that of Approach I. Thus a large number of adders have to be allocated to the higher stages and so the area efficiency will be reduced. We have found that Approach II is very suitable for multipliers whose length is  $n \leq 11$ .

Except for  $9 \times 9$  bit and  $11 \times 11$  bit multipliers, we have found that all cross-stage interconnections can be eliminated by both approaches.

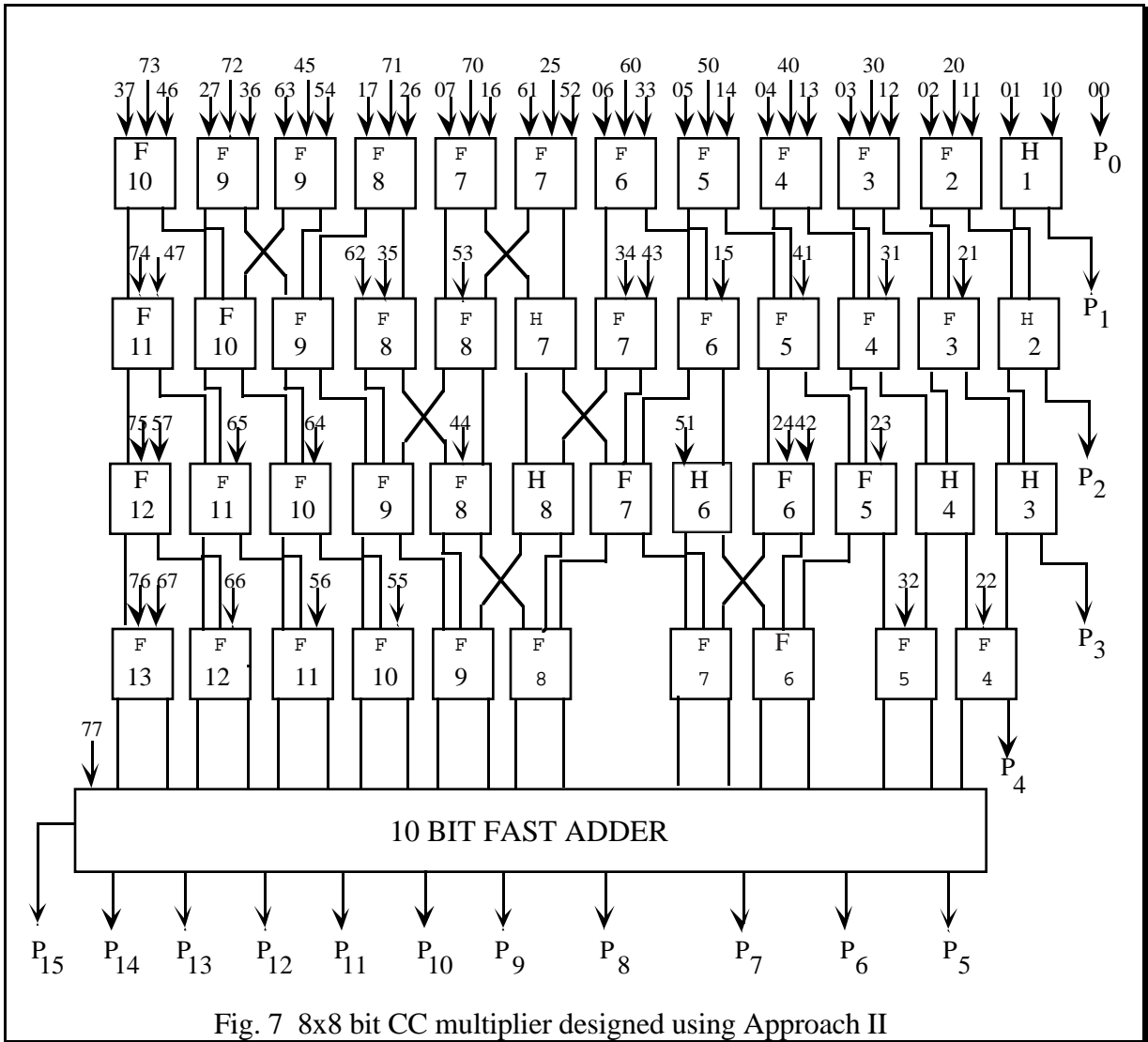
Table 3 The Maximum Number of Adders for Each Stage for Approach II

Stage	$N_m(k)$	$\hat{N}(k)$	condition
1	$2n-2-K$	$2n-2-K$	$n > 4$
2	$3n-4$	$5n-6-K$	$n > 6$
3	$2(2n-3-K)$	$8n-10-4K$	$n > 9$
4	$3(2n-4-K)$	$14n-22-7K$	$n > 13$

Table 3 lists the maximum number of adders for each stage using approach II for short length

multipliers.

Fig. 7 shows the design of the 8x8 multiplier using Approach II.



As with the design in Fig. 6; all interconnections are either to the nearest or to the next nearest neighbor and there are no cross-stage interconnections. The new design raises the area efficiency from 87.5% to 97.5%. In addition, the length of the final adder has been reduced from 14 to 10.



the maximum column size of an unsigned multiplier. If we strictly follow series (2), then the size of the multiplier,  $n$ , is related to the required number of stages,  $K$ , by:

$$\sigma(K - 1) < n + 2 \leq \sigma(K) \quad (18)$$

For example, the maximum column size of an  $8 \times 8$  two's complement multiplier is 10, and following (2), the CC part of the multiplier would require five stages.

Since the size of the largest column is three larger than the size of its two neighboring columns, we can replace (18) by:

$$\sigma(K - 1) < n + 1 \leq \sigma(K) \quad (19)$$

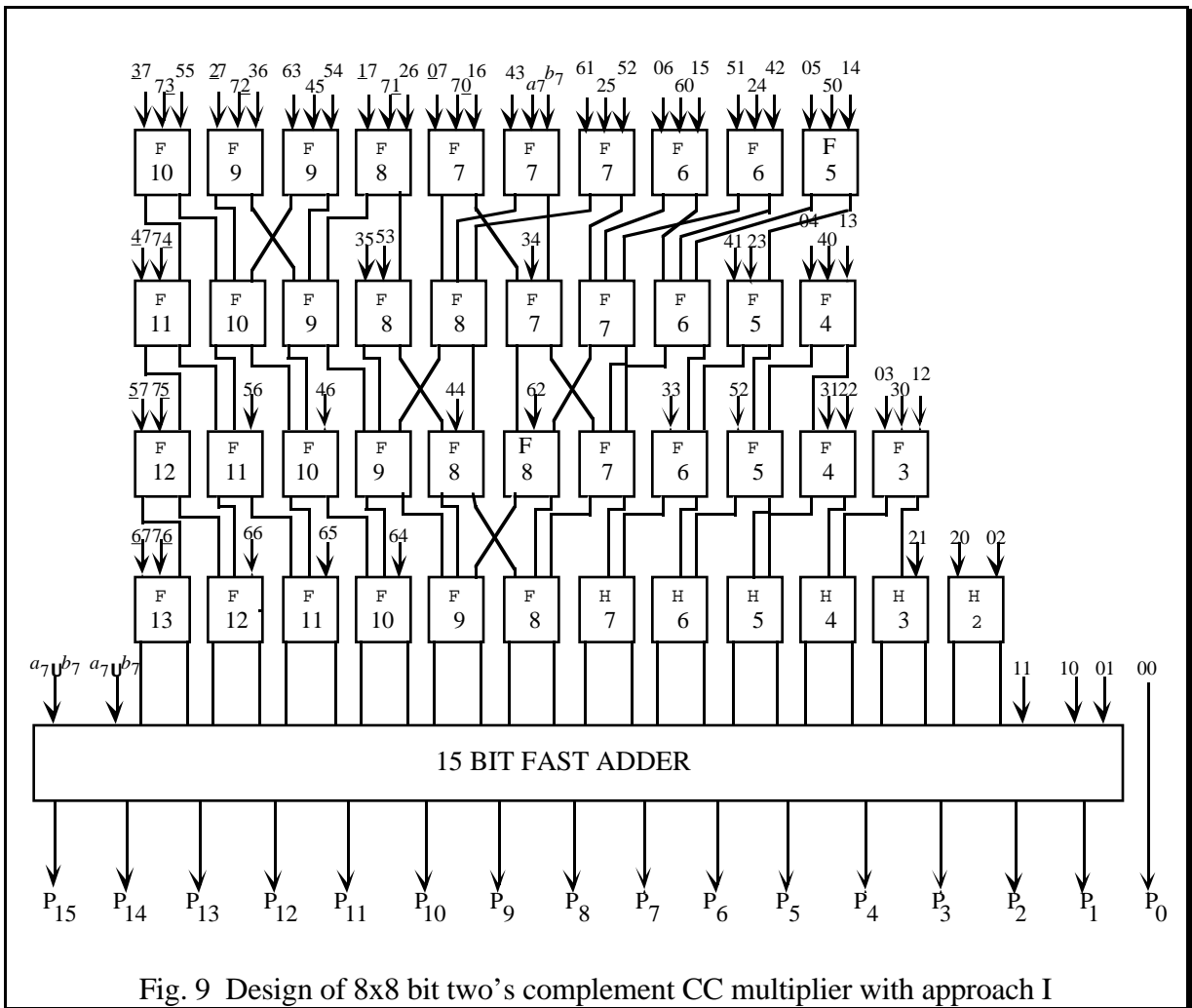
For example, the column part of an  $8 \times 8$  two's complement multiplier can be implemented in four stages, as will be shown later.

Following the same procedure given in section 2, an  $n \times n$  bit two's complement multiplier requires  $n^2 - n + 1$  adders, one more than the conventional unsigned multiplier, together with an EXCLUSIVE OR (XOR), which is located at column  $2n-1$ . Since at least one of the two entries in column  $2n-1$  is zero, there is no carry to propagate to column  $2n$ , and an XOR suffices to produce the most significant bit at column  $2n-1$ . Among the  $n^2 - n + 1$  adders, there are  $n-1$  half adders from column 1 to column  $n-1$ . The number of adders required by column  $j$  is given by

$$\begin{aligned} q_0(j) &= j & 1 \leq j \leq n-2 \\ q_0(2n-j-1) &= j & 1 \leq j \leq n \end{aligned} \quad (20)$$

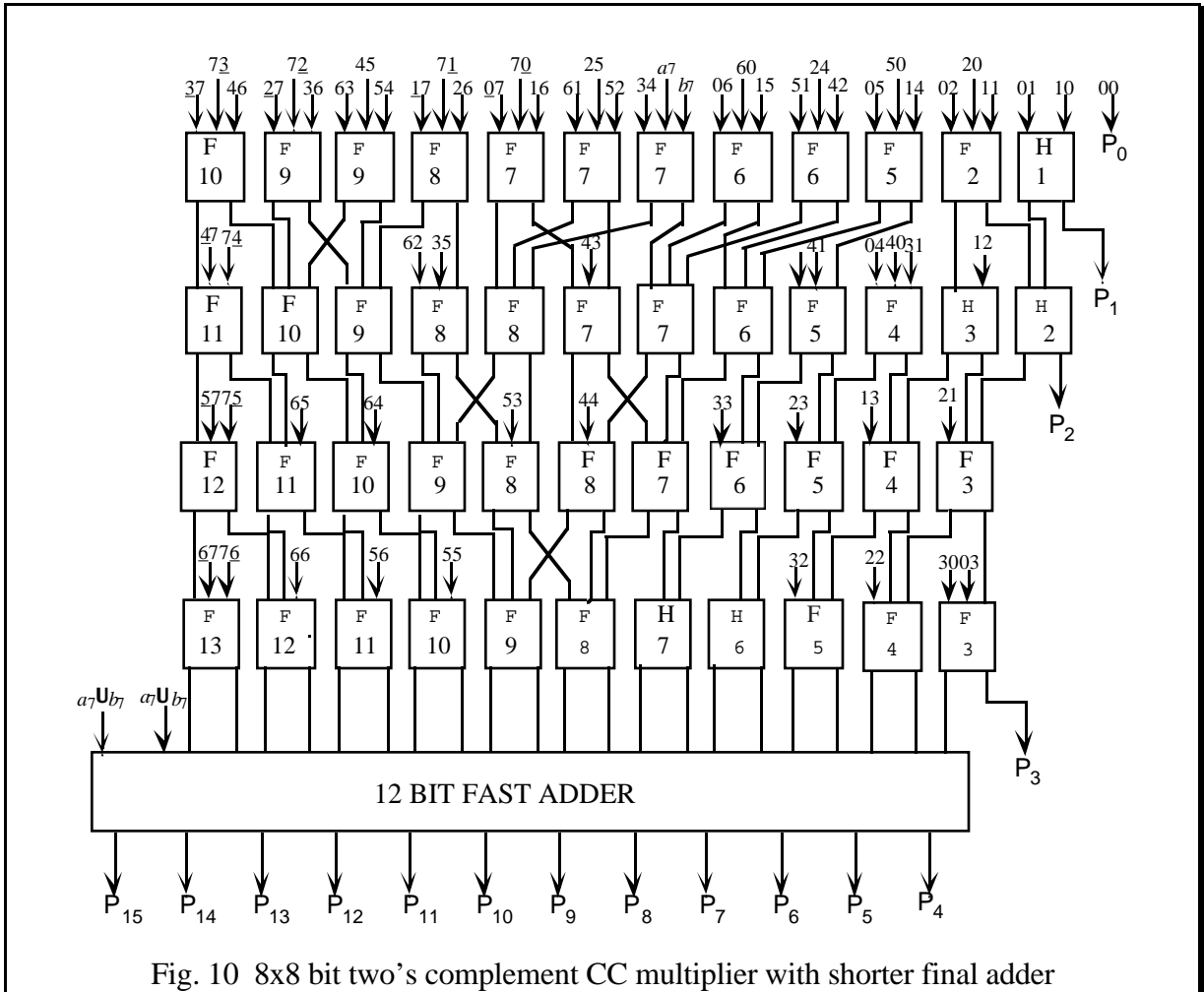
We now allocate adders to different stages observing the constraints. The XOR in column  $2n-1$  can be included in the final fast adder. In the CC part of the two's complement multiplier only one more adder, at column  $n-1$ , needs to be allocated compared to the unsigned multiplier. One simple way is to start from the architecture of the unsigned multiplier and include the extra adder in column  $n-1$ . For example, starting from Fig. 6 (Approach I), we

obtain the  $8 \times 8$  bit two's complement multiplier architecture shown in Fig. 9. where the underlined number indicates the complement. For example, 07 stands for  $\bar{a}_0b_7$ , 73, stands for  $a_7\bar{b}_3$ , etc. Because of the additional XOR, the length of the final adder is one bit longer than that for the unsigned multiplier. Most features of the unsigned multiplier design of Fig. 6 are retained with the addition of three intra-stage interconnections that are not to the nearest or next nearest neighbor.



In designing the  $8 \times 8$  two's complement CC multiplier using Approach II, the additional adder in column 7 can considerably reduce the area efficiency if the length of the final adder is only reduced by 4 (the number of CC stages). In order to achieve a higher area efficiency, we can

reduce the final adder by an extra bit, as shown by Fig. 10. The area efficiency for the CC part of this multiplier is  $46/48 = 95.8\%$ , while the length of the final adder is only 3 bits shorter than the final adder of Fig. 9. Again, except for three intra-stage interconnections that are not to the nearest or next nearest neighbor, all features of the multiplier given by Fig. 7 are retained.



## 8. Conclusions

In conclusion, we have presented two approaches to the design of CC multipliers for both unsigned and two's complement multiplication. Our approaches do not use classical restrictions

on column height sequence, and provide flexibility for distribution of adders to different stages. The number of adders used for our first approach is identical to that obtained using classical design techniques, but our technique allows more flexibility in adder placement and yields higher area efficiency. The second approach yields, in most cases for short length multipliers, higher area efficiency by reducing the length of the final adder, while maintaining the major characteristics of the first approach. In comparison with conventional design techniques, both of our design approaches increase the regularity and reduce interconnection lengths of the multiplier layout.

## 9. Acknowledgments:

The authors acknowledge support from the Natural Science and Engineering Research Council of Canada, and the Micronet Network of Centres of Excellence for funding this work. The authors also wish to thank Prof. L. Dadda for his suggestion of including two's complement multiplier designs within our new technique.

## 10. References

- [1] A. Habibi and P. A. Windy: Fast multiplier, *IEEE Trans. Comput.*, vol. C-19, pp. 153-157, 1970.
- [2] J. Y. Lee, H. L. Garvin and C. W. Slayman: A high-speed high-density silicon 8x8-bit parallel multiplier, *IEEE J. Solid-State Circuits*, vol. SC-22, 35-40, 1987.
- [3] S. Nakamura: Algorithms for iterative array multiplication, *IEEE Trans. Comput.* vol. C-35, 713-719, 1986.
- [4] S. Nakamura and K.-Y. Chu: A single chip parallel multiplier by MOS technology, *ibid*, C-37, 274-282, 1988.
- [5] B. Maden and C. G. Guy: Parallel architectures for high speed multipliers, *Proc.*

ISCAS'89, pp. 142-145, 1989.

- [6] Z. Wang, G. A. Jullien and M. C. Miller, An architecture for parallel multipliers, 25th Asilomar Conference on Signal, System and Computers, Nov. 6-8, 1991, Pacific Grove, CA
- [7] Y. Ofman: On the complexity of discrete functions, Soviet Physics-Doklady, vol. 7, pp.589-591, 1963.
- [8] C. S. Wallace: A suggestion for a fast multiplier, IEEE Trans. Electronic Computers, vol. EC-13, pp. 14-17, 1964.
- [9] L. Dadda: Some schemes for parallel multipliers, Acta Frequenza, vol. 45, pp. 574-580, 1966.
- [10] P. R. Cappello and K Steiglitz: A VLSI layout for a pipe-lined dadda multiplier, ACM Trans. Comp. Syst., pp. 157-174, 1983.
- [11] D. G. Crawley and G. A. J. Amaratunga: 8x8 bit pipelined Dadda multiplier in CMOS, IEE Proc. vol. 135 Pt. G, pp. 231-240, 1988.
- [12] M. T. Santoro and M. A. Horowitz: SPIM: A pipelined 64x64 bit iterative multiplier, IEEE J. Solid-State Circuits, vol. SC-24, pp. 487-494, 1989.
- [13] M. Nagamatsu et al: A 15-ns 32x32 bit CMOS multiplier with an improved parallel structure, *ibid*, vol. SC-25, pp. 494-497, 1990
- [14] M. Mehta, V. Parmmar, and E. Swartzlander Jr: High-speed multiplier design using multi-input counter and compressor circuits. 1991 IEEE Conf. on Computer Arithmetic.
- [15] C. R. Bouch and B. A. Wooley: A two's complement parallel array multiplication algorithm, IEEE Trans. Comput., vol. C-22, pp. 1045-1047, 1973.
- [16] P. E. Blankenship: Comments on "A two's complement parallel array multiplication algorithm", IEEE Trans. Comput., vol. C-23, p. 1327, 1974.

# *Appendix A*

## Glossary of Terms

CC	Column compression
$e(j)$	Number of bits to be added in column $j$ (partial products plus carries)
$J_k$	The highest adder index for stage $k$
$K$	Required number of stages for column compression
$n$	Length of the multiplier and multiplicand in bits
$N$	Total number of adders for column compression
$\bar{N}_k$	Average number of adders excluding the first $k - 1$ stages
$N(k)$	Number of adders in stage $k$
$\hat{N}(k)$	Number of adders $\sum_{i=1}^k N(i)$ , up to stage $k$ , that allows the use of $N_{\max}(k + 1)$ adders in stage $k - 1$
$N_{\max}$	Upper bound on number of adders in stage $k$
$p(j)$	Number of partial products in column $j$
$q(j)$	Number of adders required in column $j$ for the CC structure
$q_0(j)$	Number of necessary adders required to compute an $e(j)$ bit addition in column $j$
$q_k(j)$	Number of adders required in column $j$ of stage $k$
$s_k(j)$	Number of slots provided for stage $k$ by previous stages in column $j$
$\sigma(j)$	Stage index for the column compression part of the multiplier

## Preferred Address for Proofs

Dr. G.A. Jullien

Dept. Electrical Engineering

University of Windsor

401, Sunset Ave.

Windsor, Ontario, Canada N9B 3P4

## Figure Captions

Fig.1 Typical structure and interconnections for:

(a) Dadda multiplier; (b) 4-2 compressor multiplier; and (c) (7,3) counter multiplier

Fig. 2 (a) Partial product matrix of an 8x8 multiplier, (b) An alternative representation

Fig. 3 Expansion of column size by carries and distribution of full and half adders

Fig. 4. Table layout of the CC part of an  $8 \times 8$  bit CC multiplier

Fig. 5 A new area efficient design for an 8x8 CC multiplier

Fig. 6 Design of 8x8 bit CC multiplier with minimized interconnection length

Fig. 7 8x8 bit CC multiplier designed using Approach II

Fig. 8 Partial product matrix of an 8x8 two's complement multiplier

Fig. 9 Design of 8x8 bit two's complement CC multiplier with approach I

Fig. 10 8x8 bit two's complement CC multiplier with shorter final adder

## Table Captions

Table 1 The Maximum Number of Adders for Each Stage

Table 2 An example distribution of adders for an  $8 \times 8$  bit CC multiplier

Table 3 The Maximum Number of Adders for Each Stage for Approach II

## Acknowledgments:

The authors acknowledge support from the Natural Science and Engineering Research Council of Canada, and the Micronet Network of Centres of Excellence for funding this work. The authors also wish to thank Prof. L. Dadda for his suggestion of including two's complement multiplier designs within our new technique.

## Author Affiliation

Zhongde Wang, G.A. Jullien, W.C. Miller

VLSI Research Group

University of Windsor

Windsor, Ontario, Canada, N9B 3P4

## Brief Biographical Sketch

### Graham A. Jullien

Graham Jullien was educated in the United Kingdom. He was a student engineer and data processing engineer at English Electric Computers, UK, from 1961 to 1966, and a visiting senior research engineer at the Central Research Laboratories of EMI Ltd., UK, from 1975 to 1976.

Since 1969 he has been with the Electrical Engineering Department of the University of Windsor, Ontario, Canada, and currently holds the rank of University Professor. He is also Director of the VLSI Research Group at the University of Windsor. He was a member of the Board of Directors of the Canadian Microelectronics Corporation from 1990 to 1993 and is a Principle Researcher and member of the Coordinating Committee of the Micronet Network of Centres of Excellence.

He has published widely in the fields of Computer Arithmetic, Digital Signal Processing and VLSI Systems, and teaches courses in related areas. He is on the Editorial Board of the Journal of VLSI Signal Processing and is an Associate Editor of the IEEE Transactions on Computers. He hosted and was program co-chair of the 11th IEEE Symposium on Computer Arithmetic, and is Registration Chair for the 1995 International Conference on Acoustics, Speech and Signal Processing.

## Biography ...William C. Miller

William C. Miller was born in Toronto, Ontario, Canada. He received the B.S.E. degree in Electrical Engineering from the University of Michigan, Ann Arbor, in 1960, and the M.A.Sc. and Ph.D. degree in Electrical Engineering from the University of Waterloo, Waterloo, Ontario in 1961 and 1969, respectively. He joined the Department of Electrical Engineering at the University of Windsor in 1968, where he currently holds the rank of Professor. His research interests are oriented towards digital signal processing and the design of massively parallel VLSI processor architectures for application specific problems in the area of image processing relating to machine vision. He also teaches courses in the area of circuit theory, signal processing and system theory. Dr. Miller is also engaged extensively in industrial consulting work. He was director of the CAD/CAM centre at the University of Windsor for a two year period ending in 1988. Dr. Miller is a registered Professional Engineer in the Province of Ontario, and a member of the IEEE.