# ON DESIGN AND PERFORMANCE OF VLSI BASED PARALLEL MULTIPLIER

Dharma P. Agrawal, Girish C. Pathak, Nikunja K. Swain and Bhuwan K. Agrawal

Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, NC 27650

ABSTRACT

This paper introduces the VLSI design and layout of a $(\log^2 n)$ time n-bit binary parallel multiplier for two unsigned operands. Proposed design consists of partitioning the multiplier and multiplicand bits into four groups of n/4 bits each and then reducing the matrix of sixteen product terms using three to two parallel counters and Brent-Kung (log n) time parallel adder. Area-time performance of the present scheme has been compared with the existing schemes for parallel multipliers. Regular and recursive design of the multiplier is shown to be suitable for VLSI implementation and an improved table look up multiplier has been used to form the basis of the recursive design scheme.

INDEX TERMS: Area Time Complexity, Parallel Counters, Parallel Multiplier, Partitioning, Recursive Scheme, Table Lookup Multiplier.

## I. INTRODUCTION

Multiplication of two binary numbers consists of generating partial products and adding partial products to the shifted sum of all previous results. This shift and add of partial products to perform a multiplication of two binary numbers takes a long time. In order to speed up the multiplication operation an elegant sequential algorithm was invented by Booth [3] which generates the partial products that are +1, 0 and -1 times the multiplicand. Here the speed up (which is a factor of two on the average) is achieved in terms of generation of fewer partial product terms. Booth's algorithm was further modified for quaternary scheme [14] to improve the multiplication speed. The sequential multipliers [3,14] offer simplicity of logic and lower cost as compared to parallel multipliers but their speed of computation is lower than the parallel multipliers. So, when it comes to the applications such as real-time signal processing, where the speed of computation is the crucial factor, the use of parallel multipliers becomes essential.

Several schemes for parallel multipliers [1,8,9,10,11,13,15] have been proposed by several authors and these schemes can be classified in to two categories -- iterative arrays [1,9] and reduction type [8,13,17] in which a matrix of partial product terms is subjected to reduction by means of parallel counters. In this paper we present the design of a n-bit parallel multiplier to multiply two unsigned operands based on the reduction principle. This proposed recursive scheme for multiplication modifies Luk's [11] parallel multiplier to incorporate larger partitions (greater than two) of the multiplier and multiplicand bits, uses Brent's [4] adder and Stenzel's [15] parallel counter for reducing the partial product matrix. It takes $(\log^2 n)$ time to perform n-bit multiplication. VLSI optimality [7] i.e. (the area x time) complexity of the multiplier has been compared with the existing parallel multipliers and a VLSI layout of the proposed multiplier has been presented. An improved table look-up technique has been suggested to form the basis (lowest level multiplier) of our n-bit multiplier.

## II. n-BIT MULTIPLIER

Luk's [11] multiplier is based on the fact that the multiplication of two n-bit numbers can be accomplished by four n/2 bit multiplications, one n-bit addition and a single 2n-bit addition. Consider X,Y be two n-bit binary numbers (assuming that n is a power of 2) and say a,c, be the most significant n/2 bits and b,d be the least significant n/2 bits of x and y, respectively. Then product XY can be expressed as:

$$XY = (2^{n/2} \cdot a+b)(2^{n/2} \cdot c+d)$$

or

$$XY = 2^n ac + 2^{n/2} (ad+bc) + bd \qquad (1)$$

In this equation ac, ad, bc and bd consist of four n/2 bit multiplications and each of them result in a n bit long number. $2^n ac$ and bd form a 2n-bit number and it is added to the sum (n-bit addition) of ad and bc, with appropriate shifts, by a 2n-bit adder. This n-bit multiplication could be recursively reduced to 2x2 bit or 4x4 bit multiplication level where it could be realized by PLA or ROM multiplier.

In our scheme as opposed to Luk's [11] scheme we partition the multiplier and multiplicand bit

into four groups. This gives sixteen partial product terms. Section IV discusses the gain in the speed (as compared to Luk's scheme) achieved by the method of grouping the matrix of sixteen terms and using Brent-Kung's [4] O(log n) parallel adder and three to two parallel counters (these counters have been shown to be optimum by Reusens et. al [13].). Aho et. al. [2] also suggest a possible method of designing an asymptotically faster (sequential) algorithm if one is able to express the binary multiplication (in the case of four partitions) in terms of eight or fewer multiplications as compared to sixteen in the normal procedure. Theorem 1 proves that it is not possible to devise such an algorithm.

Theorem 1: Multiplication of two n-bit numbers cannot be expressed in terms of eight or less smaller multiplications of size O(n/4).

Proof: Consider two n-bit binary numbers X and Y where $x=4^K$ and K is an integer greater than 0. Let $x_1,x_2,x_3,x_4$ and $y_1,y_2,y_3,y_4$ be the four partitions of X and Y, respectively, such that

$$X = x_4.2^{3n/4} + x_3.2^{n/2} + x_2.2^{n/4} + x_1$$

and

$$Y = y_4.2^{3n/4} + y_3.2^{n/2} + y_2.2^{n/4} + y_1$$

Partial product term matrix of product XY consists of sixteen terms as shown in Fig. 1a. It is clear that terms $x_1 \cdot y_1$ and $x_4 \cdot y_4$ have to be achieved through normal n/4 bit multipliers. This enables us to achieve the sums of second and sixth column by (n/4+1) bit multiplications, i.e. $(x_2+x_1)(y_2+y_1)$ and $(x_4+x_3)(y_4+y_3)$ and followed by subtractions of $(x_1 \cdot y_1+x_2 \cdot y_2)$ and $(x_3 \cdot y_3)+(x_4 \cdot y_4)$ respectively, and preceded by n/4 bit addition only if we have partial products $x_2 \cdot y_2$ and $x_3 \cdot y_3$. So we need product terms $x_2 \cdot y_2$ and $x_3 \cdot y_3$. Now we can also achieve the terms of third and fifth column by multiplications $(x_4+x_2)(y_4+y_2)$ and $(x_3+x_1)(y_3+y_1)$ followed by two subtractions of $x_2 \cdot y_2$, $x_{4y} \cdot y_4$ and $x_3 \cdot y_3$, $x_1,y_1$ respectively. Product terms of the fourth column are yet to be achieved and would involve one more multiplication, but this would amount to a total of nine multiplications. This means that an n-bit multiplication cannot be performed by eight or less multiplications of order n/4.

In our scheme 16 product terms are generated in parallel. We divided the parallelogram of product terms into two symmetrical groups as shown in Fig. 2. In this figure each $a_i$, i=1 to 16 represents one partial product term of Fig. 1a and is n/2 bits long. The bit positions of the $a_i$'s are shown in Fig. 1b. Each group is separately added to produce (5n/4+1) bit long number. Adding the result of these two groups by a 2n-bit adder we get the product XY. In group 1 (group 2) product terms of third column (fifth column) are fed to a three to two counter. One of the outputs of three to two counter is appended with product term of first column (seventh column) to form n/2 bit long number and then added to other output of the three to two counter by n+1 bit adder (n bit adder in group 2).

The difference between our scheme and Luk's scheme lies in the way partial product terms are reduced. In Luk's scheme (when n-bit multiplication is expressed in terms of n/4 bit) the matrix of partial product terms is divided into four groups as shown in Fig. 3 and each group is added separately. Whereas in our scheme the matrix is divided diagonally into two equal size groups as shown in Fig. 2. Section IV shows as to how the use of counters enables to increase the speed of multiplication.

## III. LAYOUT OF AN N-BIT MULTIPLIER

Fig. 4 presents the detailed layout of an n-bit multiplier. This n bit multiplier is suitable for VLSI implementation as it is a simple recursive structure. The multiplier has 2n number of input lines where n is the number of bits in each operand. These input lines are fed into sixteen n/4 bit multipliers through a shuffle network which is not shown in Fig. 4. Outputs of the sixteen multipliers are divided into two groups in accordance to Fig. 2. Each group consists of eight product terms. In group 1, $a_2$ and $a_4$ form a n bit number and so do $a_5$ and $a_7$. These two numbers are added in a n-bit CLA. Partial product terms $a_3$, $a_6$ and $a_9$ are fed to a n/2 bit three to two counter. Sum output of the counter is appended with partial product term $a_1$ and carry output of the counter is padded with zeros in the least significant n/2 positions and thus formed these two numbers are added by a (n+1) bit CLA. The outputs of both the CLAs are added togather through (5n/4+1) bit CLA. Similarly group 2 output is achieved through adders and counters. Outputs of both groups are fed to a 2n bit carry look ahead adder to produce 2n bit output of the multiplier. In the layout all the adders are Brent-Kung type [4]. The layout is recursively drawn to the lowest level where multipliers are formed by a table look up technique and it has been discussed in Section V. Each adder is preceded by a shuffle network to position the inputs appropriately. It is easy to see that two layers of metalization [12] can replace aforesaid shuffle networks.

## IV. PERFORMANCE OF THE MULTIPLIER

This section presents the performance evaluation of the proposed multiplier. We compare our scheme with that of Luk's and present a comparison table with various other existing schemes. Evaluation criteria are based upon 1) computation speed, and 2) area time complexity for VLSI implementation.

Let

$T_m(n)$ be n-bit multiplication time
$T_a(n)$ be n-bit addition time
$A_m(n)$ area taken by n-bit multiplier
$A_a(n)$ area taken by n-bit adder.

The computation speed of our scheme can be derived from Fig. 2 and is given by:

$$T_m(n) = T_m(n/4) + T_a(n) + T_a(1) + T_a(5n/4+1) + T_a(2n) \quad (2)$$

In this equation $T_a(1)$ is the time taken by three to two counter of n/2 size which is simply equal to one addition time.

In the Luk's scheme the n-bit multiplication time when expressed in terms of n/4 multiplication, would be

$$T_m(n) = T_m(n/4) + T_a(n/2) + T_a(n) + T_a(n) + T_a(2n) \quad (3)$$

Equation (2) and (3) gives the time taken by our scheme and Luk's scheme respectively. Since $(T_a(1) + T_a(5n/4+1)) < (T_a(n) + T_a(n/2))$ our scheme will have higher computation speed (for larger value of n).

Simplifying equation (2) we get

$$T_m(n) \cong T_m(n/4) + 2T_a(n) + T_a(2n) \quad (4)$$

Using Brent-Kung's parallel adders of O(logn) time and O(nlogn) area we get

$$T_m(n) = O(\log^2 n) \quad (5)$$

Area requirement of our scheme can be derived from Fig. 4 and is given by

$$A_m(n) = 2A_a(n) + 2A_a(n/2) + A_a(n+1) + A_a(n) + 2A_a(5n/4+1) + A_a(2n) + 16A_m(n/4) \quad (6)$$

In the Luk's scheme

$$A_m(n) = 16A_m(n/4) + 4A_a(n/2) + 4A_a(n) + A_a(n) + A_a(2_n) \quad (7)$$

Since $2A_a(n/2) \cong A_a(n+1)$ there is no significant real estate value (area complexity) difference in Luk's and our scheme.

Further applying Brent-Kung's [4] parallel adder area requirement we get the area for our multiplier as

$$A_m(n) = O(n^2 \log n) \quad (8)$$

Table I shows the performance comparison table of various other schemes with our scheme.

## V. TABLE LOOK UP BASIS

So far we have not discussed the level at which the recursion would stop in our scheme and the type of multiplier at that level. Table look up or ROM multiplier is the simplest choice. But it requires a very large area. Conventional table look up n bit multiplier consists of a memory of $2^{2n}$ words with each word of length 2n. Concatenated bits of multiplicand and multiplier numbers form the address of the memory which maintains the multiplier output of these two numbers. The following technique which would reduce memory size to $2^n$ for an n-bit multiplier has been used to form the basis of our multiplier.

Consider X and Y be two n-bit number and we know that

$$XY = 1/4\ ((X+Y)^2-(X-Y)^2) \quad (9)$$

This implies that multiplication can be expressed in terms of addition/subtraction and squaring operations. X+Y and X-Y are n+1 bit and n-bit long, respectively. Division by four is shifting by two positions toward the right. We need a squaring table for (n+1) long number and n-bit long number. Since multiple read type memories [6] do exist so we can have a squaring table for n+1 bit having two read port for parallel access. Shifting is avoided by having the memory word of size 2n instead of (2n+2) for (n+1) bit squaring table.

So the multiplication has been changed to two parallel addition/sub followed by two square table access and followed by a subtraction.

This technique although requires two addition/sub time more than conventional table look up but saves a considerable area (of size $2^n$). Also we have lower memory access time because of smaller memory size. The level at which such table look up multiplier will be placed can be found by optimizing various delays, computation time and area requirement of computing elements.

## VI. DISCUSSION

In this paper we have presented a design and performance evaluation of an n-bit binary parallel multiplier of $O(\log^2 n)$ time. As the time performance of any circuit largely depends upon geometry rather than the topology [7], we have employed parallel counters to provide an alternative way of reducing the partial product terms in Luk's scheme. We have used a table look up technique which requires $O(2^n)$ size chip area as opposed to the conventional table look up of size $O(2^{2n})$ for n-bit binary multiplication. We have used a partition of size four and higher order partitions may improve the multiplier speed.

REFERENCES


[1] D.P. Agrawal, "High speed arithmetic arrays," IEEE Trans. on Comput., Vol. C-28, No. 3, Mar. 1979, pp. 215-224.

[2] A.V. Aho, J.E. Hopcroft and J.D. Ullman, The design and analysis of computer algorithms, Addison-Wesley, Reading, Massachusetts, 1974.

[3] A.D. Booth, "A signed binary multiplication technique," Quart. Journ. Mech. and Applied Math., vol. IV, Pt. 2, 1951, pp. 236-240.

[4] R.P.Brent and H.T. Kung, "A regular layout for parallel adders," Technical report, Dept. of Comput. Science, Carnegie-Mellon University, CMU-CS-79-131, June 1979.

[5] R.P. Brent and H.T. Kung, "The area-time complexity of binary multiplication," Technical report, Dept. of Comptuer Science, Carnegie Mellon University, July 1979.,

[6] S.S.L. Chang, "Multiple read single write memory and its applications," IEEE Trans. on Comput., Vol. C-29, No. 8, Aug. 1980, pp. 689-694.

[7] B. Chazelle and L. Monier, "Optimality in VLSI," Dept. of Comput. Science, Carnegie-Mellon University, CMU-CS-81-141, Sept. 1981.

[8] L. Dadda, "On parallel digital multipliers," Altra Frequenza, Vol. 45, No. 10, Oct. 1976, pp. 574-580.

[9] H.H. Guild, "Fully iterative fast array for binary multiplication and fast addition," Electron, Lett., Vol. 38, May 1969, pp. 843-852.

[10] A. Habibi and P.A. Wintz, "Fast multipliers," IEEE Trans. on Comput., Vol. C-19, Feb. 1970, pp. 153-157.

[11] W.K. Luk, "A regular layout for parallel multiplier of $O(\text{Log}^2 N)$ time," CMU Conference on VLSI systems and computations, Oct. 19-21, 1981, Pittsburg, Penn., pp. 317-326.

[12] C. Mead and L. Conway, Introduction to VLSI systems, Addison-Wesley, Reading, Mass., 1980.

[13] P. Reusens, W.H. Ku and Y.H. Mao, "Fixed point high speed parallel multipliers in VLSI," CMU Conference on VLSI systems and computations, Oct. 19-21, 1981, Pittsburgh, Penn., pp. 301-310.

[14] L.P. Rubinfield, "A proof of modified Booth's algorithm for multiplication," IEEE Trans. on Comput., Vol. C-24, Oct. 1975, pp. 1014-1015.

[15] W.J. Stenzel, W.J. Kubitz and G.H. Garcia, "A compact high speed parallel multiplication scheme," IEEE Trans. on Comput., Vol. C-26, No. 10, Oct. 1977, pp. 948-957.

[16] N.R. Strader and V.T. Rhyne, "Canonical bit sequential multiplier," accepted for publication in the IEEE Trans. on Comput., Feb. 1983.

[17] C.S. Wallace, "A suggestion for a fast multiplier," IEEE Trans. on Electronic Comput., Vol. EC-13, Feb. 1964, pp. 14-17.


Table I Performance comparison of various schemes for parallel multiplier

| Multiplier type | Time | Area-Time Product |
|---|---|---|
| Brent-Kung's | $O(n^{1/2}\log n)$ | $O(n^2 \log^3 n)$ |
| Reusens' | $O(n)$ | $O(n^3)$ |
| Proposed scheme | $O(\log^2 n)$ | $O(n^2 \log^6 n)$ |
| Aho et. al's. | $O(\log^2 n)$ | $O(n^2 \log^4 n)$ |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | $x_4$ | $x_3$ | $x_2$ | $x_1$ | |
| | x | $y_4$ | $y_3$ | $y_2$ | $y_1$ | |
| | | | | $x_4y_1$ | $x_3y_1$ | $x_2y_1$ | $x_1y_1$ |
| | | $x_4y_2$ | | $x_3y_2$ | $x_2y_2$ | $x_1y_2$ | |
| | $x_4y_3$ | $x_3y_3$ | | $x_2y_3$ | $x_1y_3$ | | |
| $x_4y_4$ | $x_3y_4$ | $x_2y_4$ | | $x_1y_4$ | | | |

Fig. 1a Partial product terms of xy.

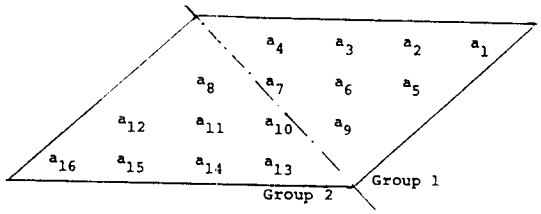


Fig. 2 Partitioning in the proposed scheme.

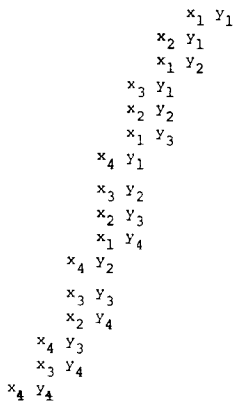


Figure 1b Bit positions of $a_i$, i=1 to 16

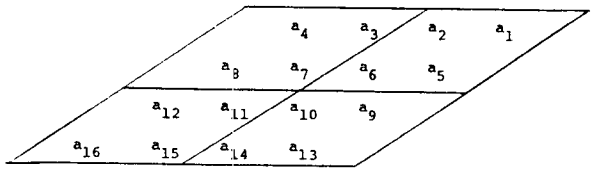


Fig. 3 Partitioning in the Luk's scheme.

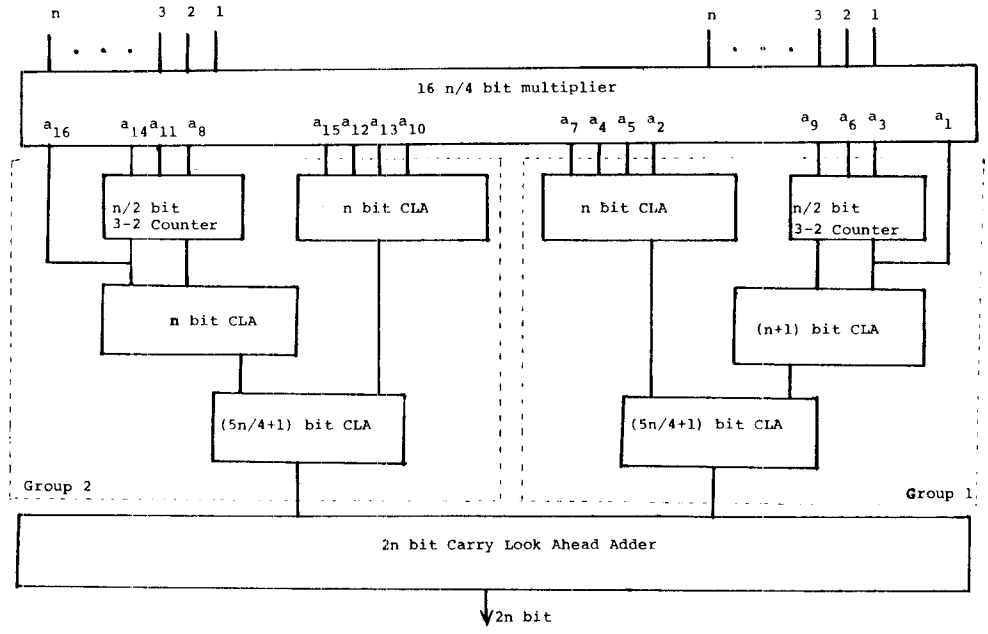


Fig. 4 n bit Multiplier Layout