On Binary Multiplication Using the Quarter Square Algorithm

TOTADRI JAYASHREE AND DHRUBA BASU

***Abstract*—This correspondence suggests a new method of performing binary multiplication using the quarter square technique. As compared to the previous methods [1], [2] the proposed scheme is more general in that it can be readily applied to numbers of any word length. Moreover, it has been shown that for the eight-bit case the present method is faster, more systematic, and economical than the earlier schemes.**

***Index Terms*—Binary multiplication, Dadda's scheme, quarter-square multiplier, squaring matrix.**

I. Introduction

Using the well-known quarter square algorithm [5], the problem of binary multiplication is reducible to that of squaring of binary numbers. This has been studied by Ling [1] and Chen [2]. Ling has proposed a scheme in which the product of two numbers A and B was obtained using an expression involving two terms $S(i)$ and $S(j)$; i and j being the normalized binary fractions corresponding to the sum and difference of A and B, respectively. The transfer function $S(x)$ was given by

$$S(x) = x - \frac{x^2}{2}$$

and it was shown that for an eight-bit fraction, each bit of $S(x)$ would require the ORing of at most 26 terms, each term involving not more than 7 variables. Chen has suggested a simpler method in which he obtains the square of an eight-bit binary number

$$p = 0 \cdot P_1 P_2 P_3 P_4 P_5 P_6 P_7 P_8$$

as

$$p^2 = Y + Z.$$

The analysis of the expression for Z shows that each bit of Z involves no more than 14 terms and each term involves no more than 6 variables. The bits of Y can be formed by ANDing two variables.

In this correspondence, we present an alternative approach to the squaring problem. We have shown that for an n bit number, the n row squaring parallelogram can be reduced to a $(n - p + 2)/2$ row triangular array where $p = 3$ for n odd, and $p = 4$ for n even.

The first row L of this reduced matrix is derived from the original n row matrix by combining the antidiagonal with the elements embedded in three successive layers parallel to the antidiagonal and on either side of it. This row L consists of $2n$ elements such that each element can be generated by ORing at most four terms and each term is obtained by ANDing not more than three variables. Moreover, for this row, elements belonging to all the odd columns starting from column 5 and ending with column $(2n - 3)$ have a similar logical expression and hence can be realized by the same hardware structure. Similarly, the elements in all the even columns between columns 6 and $(2n - 2)$ have identical hardware implementation.

The remaining elements in the original parallelogram are rearranged to form the rows 2 through $(n - p + 2)/2$ of the reduced matrix and each of these can be obtained by the ANDing of two variables. Further reduction of this reduced matrix to form the result can be achieved by using a combination of an adder tree and a string of full adders as suggested by Dadda [3] and [4]. In the following section, we describe a detailed development of the proposed scheme.

Manuscript received December 20, 1974; revised August 28, 1975.

The authors are with the On-Board Computer Section, Control, Guidance, and Instrumentation Division, Space Science and Technology Centre, Trivandrum, India.

II. Reduction of the Squaring Parallelogram

The problem of squaring an n bit binary number $A_n\ A_{n-1}\ A_{n-2} \cdots A_3 A_2 A_1$ is essentially one of forming the columnwise sum of the elements of the $n \times 2n$ squaring matrix shown in Fig. 1. A typical element of the squaring matrix denoted by the symbol A_{ij} can be obtained by forming the logical product of the ith and jth bits of the number, i.e., $A_{ij} = A_i \cdot A_j$.

Dadda, in his study on parallel multipliers [4], has suggested a scheme in which the original summation matrix is transformed by means of adders into a matrix with a smaller number of rows. This matrix can further be transformed into a second matrix with still less number of rows by using another set of adders, and the process can be repeated until the original matrix is reduced to a matrix consisting of two rows only. Final addition of these two rows to form the result is done using a string of full adders. A similar scheme can also be applied to the elements of a squaring matrix to form the result. However, utilizing certain properties of the squaring matrix, it is possible to reduce the matrix to one with a smaller number of rows without using adders. This results in a significant saving of hardware without any resultant loss of regularity. Dadda's method of reduction using an adder tree and a string of full adders can now be applied to this reduced squaring matrix to form the result. Now, we will describe certain properties of the squaring matrix.

Property 1: The antidiagonal R_1 (Fig. 1) dichotomizes the matrix into two symmetric halves and hence the entire matrix can, in effect, be represented by the antidiagonal and all the elements of one half of the matrix shifted by one position to the left relative to the antidiagonal [2]. Thus, the elements contained in the subdiagonals R_2, R_3, and R_4 (Fig. 1) and their symmetric counterparts can be rearranged as rows R_2', R_3', and R_4' with respect to R_1 as shown in Fig. 2(a).

Property 2: For the submatrix shown in Fig. 2(a), the carries into all the odd columns included between columns 5 and $(2n - 3)$ have a similar logical expression and since the columns themselves are similar, the elements of this submatrix can be summed up to form a result which has the same type of logical expression for all odd columns between columns 5 and $(2n - 3)$. Such a similarity also exists among all the even columns included between columns 6 and $(2n - 2)$.

Property 2 will become more clear from the following discussion wherein we have derived a general expression for the carry $C_{in(i)}$ into any odd column i. Let C_{jk} denote the carry from the jth column into the kth column.

From Fig. 2(a) we can write

$$C_{in(5)} = C_{45} = A_3 \cdot A_2 \cdot A_1$$

$$C_{in(6)} = C_{56} = A_3 \cdot A_2(\bar{A}_4 + \bar{A}_1) + A_3 \cdot A_4 \cdot A_1 \cdot \bar{A}_2$$

$$C_{57} = A_4 \cdot A_3 \cdot A_2 \cdot A_1$$

$$C_{67} = C_{in(6)} \cdot A_{42} = A_4 \cdot A_3 \cdot A_2 \cdot \bar{A}_1$$

$$C_{in(7)} = C_{67} + C_{57} = A_4 \cdot A_3 \cdot A_2.$$

Examination of the expressions for $C_{in(5)}$ and $C_{in(7)}$ reveals that the carries into the 5th and 7th columns are similar. Furthermore, as the columns themselves are identical, the carries out of these columns are also similar, i.e., C_{56} is similar to C_{78}, and C_{57} is similar to C_{79}. Thus, it can easily be seen that for all subsequent odd columns, i.e., 9, 11, $\cdots$, $(2n - 3)$ the carry into any odd column i can be given by the general expression

$$C_{in(i)} = A_{(i-3)/2} \cdot A_{(i-1)/2} \cdot A_{(i+1)/2}.$$

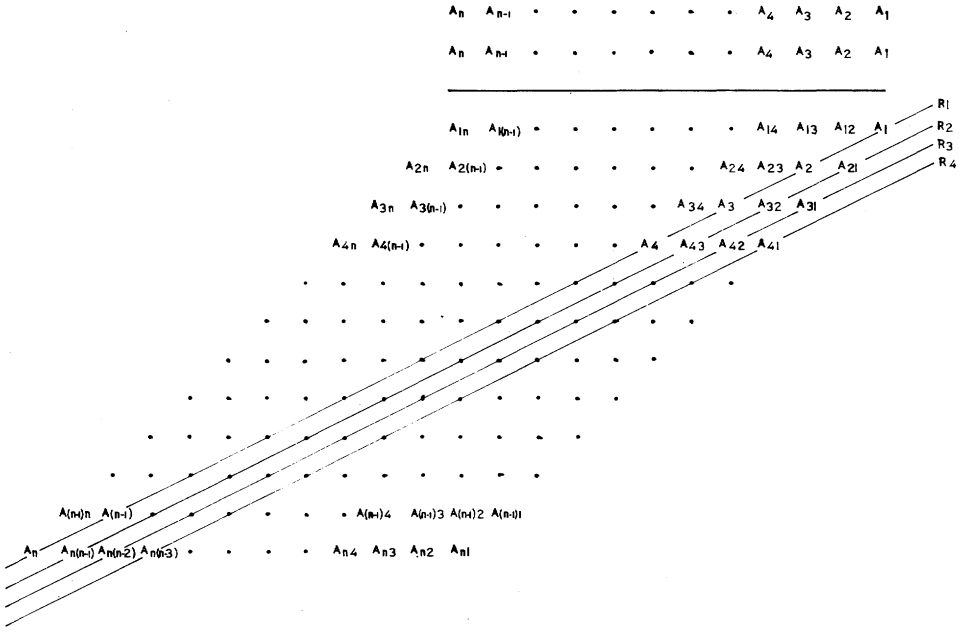


Fig. 1. The squaring matrix.

Property 2 has been used to generate the row L whose elements $L_1, L_2, \cdots, L_{2n}$ are given as follows:

$$L_1 = A_1$$

$$L_2 = 0$$

$$L_3 = A_2 \cdot \bar{A}_1$$

$$L_4 = A_1 \cdot (A_2 \oplus A_3)$$

$$L_i = A_{(i+1)/2} \cdot \bar{A}_{(i-1)/2} \cdot (\bar{A}_{(i-3)/2} + \bar{A}_{(i+3)/2}) + A_{(i-3)/2} \cdot (A_{(i+1)/2} \oplus A_{(i+3)/2}) \quad (1)$$

where $i = 5, 7, 9, \cdots, (2n-3)$

$$L_{i'} = A_{(i'-2)/2} \cdot (A_{(i')/2} \oplus A_{(i'+2)/2}) + A_{(i'-4)/2} \cdot A_{(i')/2} \cdot A_{(i'+2)/2} \quad (2)$$

where $i' = 6, 8, 10, \cdots, (2n-2)$

$$L_{2n-1} = A_n \cdot (\bar{A}_{n-1} + A_{n-2}) \quad (3)$$

$$L_{2n} = A_n \cdot A_{n-1}. \quad (4)$$

Thus, it is seen that the rows R_1, R_2', R_3', and R_4' of the submatrix can be combined without using full adders to form the row L.

The generation of the elements of row L requires only simple AND-OR logic gates. Moreover, excepting the four elements L_3, L_4, L_{2n-1}, and L_{2n}, all the remaining elements can be realized with only two different types of basic hardware. Having thus combined the rows R_1, R_2', R_3', and R_4', we can utilize Property 1 to arrange the remaining elements of the original matrix with respect to row L, to form the reduced matrix as shown in Fig. 2(b).

For any even column j of the reduced matrix such that $6 \leqslant j \leqslant (n+1)$, $A_{(j,x)}$ denotes the element in the $(x+1)$th row and is equal to $A_{(j-x)x}$ where $1 \leqslant x \leqslant (j-4)/2$, the element in the first row being given by L_j defined in (2). Similarly, for any odd column j' of the reduced matrix such that $7 \leqslant j' \leqslant (n+1)$, $A_{(j',x')}$ denotes the element in the $(x'+1)$th row and is given by $A_{(j'-x')\cdot x'}$ where $1 \leqslant x' \leqslant (j'-5)/2$, the element in the first row being given by $L_{j'}$ defined in (1). Any odd column $k'(k' > n+1)$, starting with element $L_{k'}$ defined by (1) has the element $A_{(k',y')}$ in the $(y'+1)$th row where $A_{(k',y')}$ denotes $A_{(k'-n-y'+1)\cdot(n-y'+1)}$ with $1 \leqslant y' \leqslant (j'-5)/2$, j' and k' being connected by the relation $k' = 2n + 2 - j'$. In a similar manner, any even column $k(k > n+1)$, where $k = 2n + 2 - j$ has L_k defined by (2) in the first row and the element $A_{(k,y)}$ in the $(y+1)$th row where $A_{(k,y)}$ represents $A_{(k-n-y+1)\cdot(n-y+1)}$ with $1 \leqslant y \leqslant (j-4)/2$.

The arrangement of the reduced matrix can be easily understood by considering an example where $n = 8$. This has been illustrated in Fig. 3(a). The expressions for the elements L_5 through L_{16} can be obtained from (1)–(4).

The columnwise sum of the elements of the reduced matrix can now be obtained by applying Dadda's method. The implementation of the above scheme for an eight-bit number is illustrated in the diagram of Fig. 3(b). Since squaring an n-bit number can yield at most a $2n$-bit number, there can never be a carry out of the $2n$th column. Or in other words, there can be a carry into the $2n$th column only when L_{2n} is a zero.

Hence, the adder corresponding to this column in the final adder chain can be replaced by a simple 2-input OR gate. It may also be noted that the least significant five bits of the result are given directly by the elements L_1, L_2, L_3, L_4, and L_5 of the reduced matrix.

III. Example

In order to explain the above concepts, a numerical example for $n = 8$ has been given. Let the binary numbers to be squared be $A = 1\,0\,1\,1\,0\,0\,1\,0$, for which the values of $A_1, A_2, \cdots, A_n$ are $A_1 = 0$, $A_2 = 1$, $A_3 = A_4 = 0$, $A_5 = A_6 = 1$, $A_7 = 0$, and $A_8 = 1$. Substituting these values in the expressions given for the elements of the L row we get $L_1 = L_2 = 0$, $L_3 = 1$, $L_4 = L_5 = L_6 = 0$, $L_7 = 1$, $L_8 = 0$, $L_9 = 1$, $L_{10} = L_{11} = 0$, $L_{12} = L_{13} = L_{14} = L_{15} = 1$, and $L_{16} = 0$.

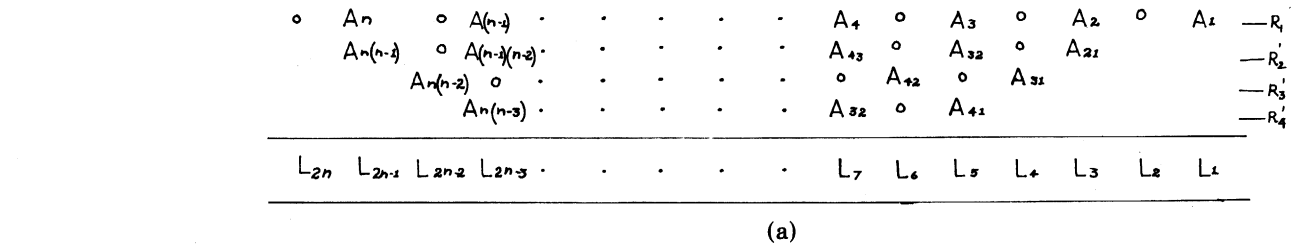
The reduced matrix for this example is given below.

0 1 1 1 1 0 0 1 0 1 0 0 0 0 1 0 0

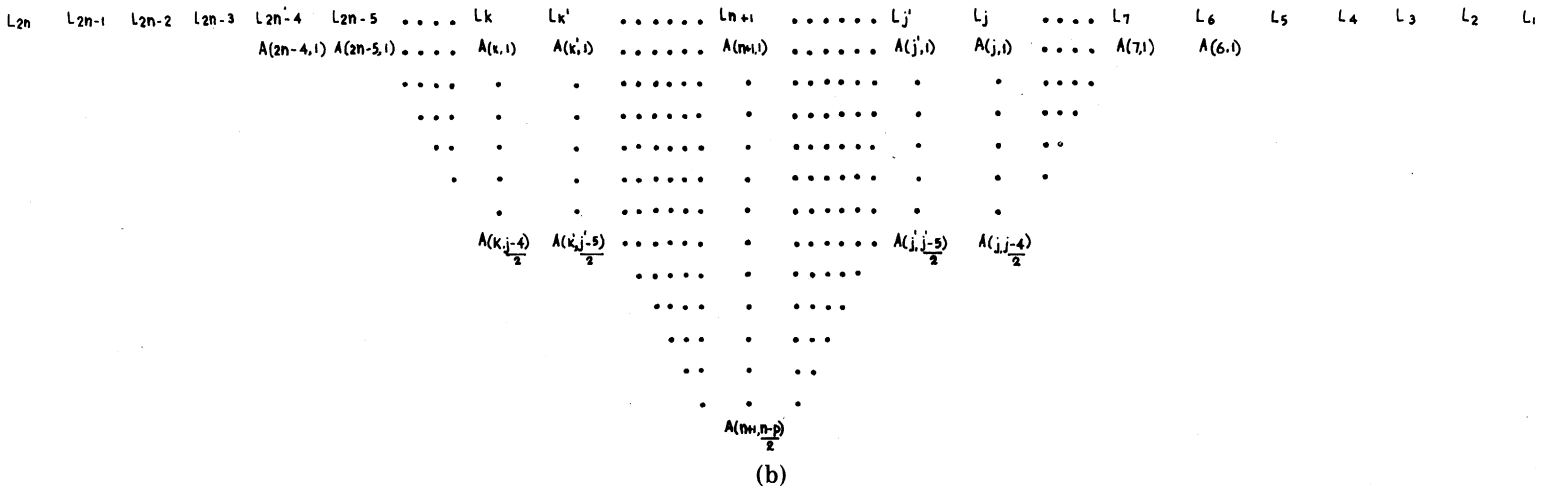
0 0 1 0 0 0 0

0 0 1

Columnwise summation of the above matrix to form the result has been shown in Fig. 3(b). It may be noted that L_1, L_2, L_3, L_4, and L_5 form the last 5 bits of the result and have not been shown in the diagram. The final result is given by $A^2 = 0\,1\,1\,1\,1\,0\,1\,1\,1\,1\,0\,0\,0\,1\,0\,0$.



(a)



(b)

Fig. 2. (a) Generation of row L. (b) The reduced matrix.

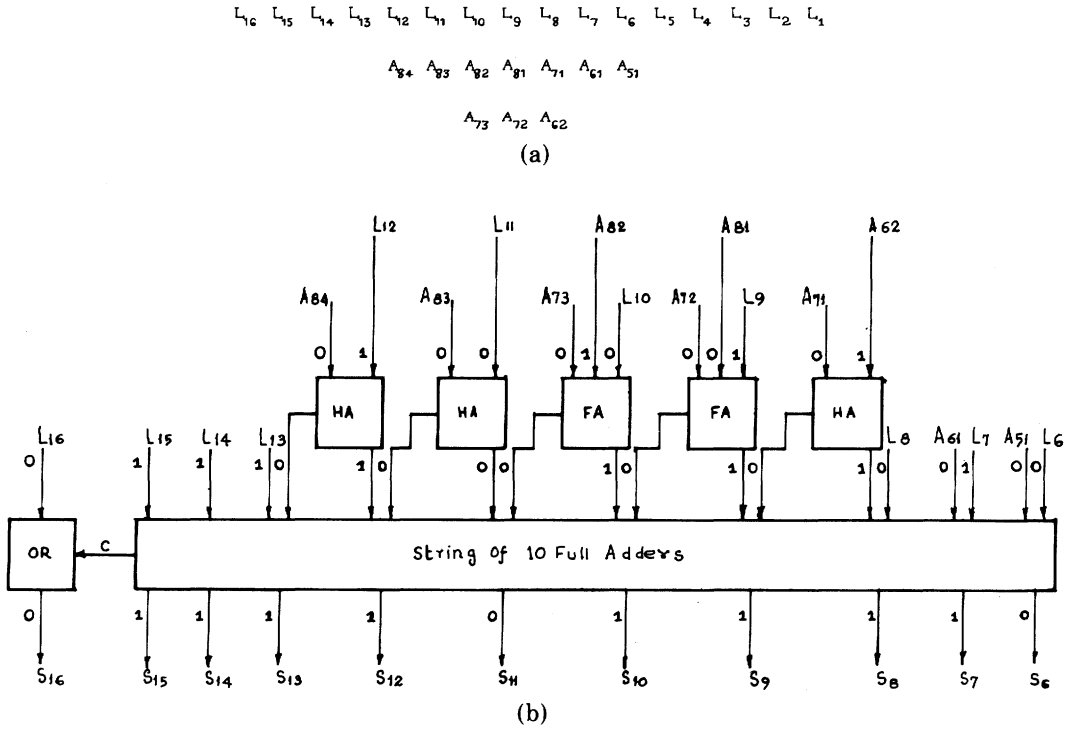


Fig. 3. (a) The reduced matrix for $n = 8$. (b) An 8-bit squaring scheme.

IV. Discussion

Choosing the particular 8-bit squaring problem given in Section III as a basis for comparing our present method with that of Chen, it is seen that our scheme is faster and much more economical. Use of large fan-in and fan-out gates is not required as the terms L_3 through L_{16} can be generated by using at most 4-input OR gates and 3-input AND gates. This may be compared to Chen's Z logic, in which the expression for z_8 would require the use of a 14-input OR gate and 6-input AND gates. Moreover, the total number of logic gates required for generating Y and Z in Chen's algorithm is 87 as compared to only 64 gates required in our case. Also, the addition of the bits of Y and Z needs 14 full adders, whereas in our scheme only 12 full adders and 3 half adders are needed to generate the result [Fig. 3(b)]. From the point of view of speed, it can be seen that Chen's method is slower because it requires a larger number of full adders in the adder string. Further, the logic for the different bits of Z is rather unsystematic, i.e., no two bits of Z can be realized by an identical hardware structure. Finally, it may be added that the schemes given by both Chen and Ling are applicable to maximum eight-bit numbers only. Any number larger than 8 bits, say 12 bits, has to be decomposed into two 8-bit blocks and the entire hardware has to be replicated 4 times for a parallel formation of the result. The major advantage of our proposed method is that it can easily be extended to any number of bits without having to decompose the number into segments of eight bits, thus resulting in a substantial saving of hardware. This is because the different bits of L as given in (1) and (2) have the same types of logical expressions for any value of n.

Acknowledgment

The authors are grateful to Dr. S. C. Gupta, Head of the Control, Guidance, and Instrumentation Division, Space Science and Technology Centre, Trivandrum, India, for his encouragement.

References

[1] H. Ling, "High speed computer multiplication using a multiple-bit decoding algorithm," *IEEE Trans. Comput.*, vol. C-19, pp. 706–709, Aug. 1970.
[2] T. C. Chen, "A binary multiplication scheme based on squaring," *IEEE Trans. Comput.*, vol. C-20, pp. 678–680, June 1971.
[3] A. Habibi and P. A. Wintz, "Fast multipliers," *IEEE Trans. Comput.*, vol. C-19, pp. 153–157, Jan. 1970.
[4] L. Dadda, "Some schemes for parallel multipliers," *Alta Freq.*, vol. 34, pp. 349–356, Mar. 1965.
[5] A. S. Jackson, *Analog Computation.* New York: McGraw-Hill, 1960, p. 477.