DESIGN FOR A RECURSIVE PARALLEL MULTIPLIER

Renato DE MORI and Régis CARDIN

Department of Computer Science
Concordia University
1455 De Maisonneuve Blvd West
Montréal,Québec,Canada
H3G 1M8

ABSTRACT

A network for performing multiplications of two two's complement numbers is proposed.The network can be implemented in a synchronous or an asynchronous way. If the factors to be multiplied have N bits, the area complexity of the network is O(N^2) for practical values of N as in the case of cellular multipliers. Due to the design approach based on a recursive algorithm, a time complexity O(log N) is achieved.

INTRODUCTION

Multipliers are fundamental components of computer arithmetic units and special purpose systems conceived, for example, for signal processing. The design of parallel multipliers has received considerable attention in the last thirty years.

A fundamental contribution to the design of combinational or simultaneous multipliers has been given by Wallace [26] who proposed a network of Carry Save Adders (CSA) for adding Partial Products (PP) generated by two integer factors represented in binary code, one with M bits, the other with N bits and obtaining in O(log MN) time two addends whose sum is equal to the product of the factors. A pipelined version of this design has been implemented in commercially available machines.

Wallace's design approach was improved by Dadda [6] who proposed to use Parallel Counters (PC) instead of CSA in order to reduce cost.

Capello and Steiglitz [5] have recently reviewed Dadda's approach and derived a pipelined multiplier that could be reduced to a simultaneous one by eliminating memory elements associated with PC outputs. This multiplier uses a network containing different types of counters and a fast carry-lookahead binary adder. This adder is based on a design by Brent and Kung [3] and adds the two numbers generated by the PC network. An adder of this type will be indicated in the following as Special Adder (SA). Assuming, for the sake of simplicity, that both the factors have N bits, the time complexity of the (SA) is O(log N) and the area complexity is O(Nlog N). The area complexity of the multiplier is O(N^2log N). Other interesting ideas along this approach are reported in [22] and [24].

At the end of the sixties, a number of multiplier designs were proposed based on iterative arrays of equal cells [9,14,15,16]. These structures have an area complexity O(N^2) and time complexity O(N). The basic cell of the iterative multipliers was a gated full-adder and "cellularity" was considered an advantage for Large Scale Integration. Using the same cells, networks for multiplying signed numbers were proposed [1,2,8,10,20] having the same area and time complexity.

In order to increase the speed of iterative multipliers having time complexity O(N), some macrocellular structures have also been proposed [7,11,12,17,23].

An extended review with interesting comments and contributions to the design of parallel multipliers can be found in the book by Hwang [18].

The purpose of this paper is that of introducing a recursive, rather than an iterative, algorithm for designing parallel multipliers. This algorithm will bring to an implementation with design advantages similar to those of iterative multipliers, but with the fundamental difference that it can be implemented by a recursive application of the same design procedure rather that by the repetition of the same cell design.

The multipliers implemented with this algorithm have area complexity O(N^2) and time complexity O(log N). This new solution combines the advantages in area complexity of the iterative multipliers and the advantages in time complexity of designs based on pseudo-additions performed by CSAs or PCs.

Making these multipliers pipelined with period complexity O(1) is easy as it is for those designed with cellular iterative networks. In this case the latency complexity becomes O(log N).

Capello and Steiglitz [5] have compared available multiplier designs using a figure of merit $F = AP^2T^2$ derived for VLSI implementation. A is the area complexity, T is the latency complexity and P is the period complexity. A lower bound, LBF, for F is reported [5] to be:

$$LBF = N^2 log^2 N \qquad (1)$$

The structure proposed by Capello and Steiglitz is the best one proposed so far for simultaneous multipliers and has a figure of merit $F = N^2 log^3 N$.

The structure proposed here has F = LBF and is the only structure proposed so far for simultaneous multipliers with $A = N^2$ and T = log N.

The structure can be adapted to multiply two two's complement numbers with the same performance, and these properties make the structure proposed here, the fastest one with O(N^2) area complexity for multiplying two's complement numbers. The design proposed here is based on moduli whose connection can be easily programmed in such a way that either

CH2146-9/85/0000/0044$01.00 © 1985 IEEE

four single precision multiplications, or one double precision multiplication can be performed under the control of some specification variables.

THE ALGORITHM FOR RECURSIVE MULTIPLIERS

In order to introduce the algorithm for designing recursive multipliers, an iterative algorithm denoted Algorithm IM for multipliying positive binary integer numbers will be first presented. This algorithm leads to the implementation of cellular multipliers with time complexity O(N). A design for a basic cell of these is proposed in [11]. With these cells, multipliers for small values of N can be designed with practically good performances. Furthermore these cellular multipliers are implemented by repeating an elementary cellular structure which is a big advantages from the point of view of automatic design, manufacturing and testing.

Algorithm IM (Iterative Multiplication)

Let

$$H = \sum_{i=0}^{N-1} h_i 2^i \qquad (2)$$

and

$$K = \sum_{j=0}^{N-1} k_j 2^j \qquad (3)$$

be the factors and

$$P = H^*K = \sum_{r=0}^{2N-1} g_r 2^r \qquad (4)$$

be the product. h_i, k_j, g_r are binary variables.

Let N = LG with L and G integers.

Wording the factors H and K in the basis 2^G one gets:

$$K = \sum_{n=0}^{L-1} K_n 2^{nG}$$

$$H = \sum_{m=0}^{L-1} H_m 2^{mG}$$

with H_m and K_n integers less then 2^G and expressible in the binary code with G bits.

The product P can be obtained with an iterative array proposed in [11].

A special cell design for G = 2 is also proposed in [11]. This cell design is based on four 16 input-1 output multiplexers and introduces a propagation delay equal to the swiching time of a multiplexer.

Let T_m be the switching time of a multiplexer, the structure proposed in [11] has a total multiplication delay.

$$T_d = NT_m \qquad (5)$$

Where N is the number of bits of the factors. The area complexity of the structure is $A_1 = O(N^2)$ and the structure can be used for adding two numbers:

$$Y = \sum_{b=0}^{N-1} y_b$$

$$X = \sum_{a=0}^{N-1} x_a$$

to the product P = H*K, making the structure capable of performing the operation :

$$Z = H^*K + X + Y \qquad (6)$$

Which is also the expression computed by the basic cell. Dean [7] has called such a structure, a ***full-multiplier***.

An algorithm has been proposed [10] for using full multipliers of positive numbers in order to multiply two's complement binary fractions with the addition of peripheral logic.

In order to reduce the time complexity from O(N) to O(logN) by keeping the area complexity approximately $O(N^2)$ a new structure, called ***recursive multiplier*** is proposed in this paper. This structure shares with the iterative multiplier proposed in [11], the following features:

-modular layout

-possibility of pipelining and using two's complement factors.

-possibility of using the structure of a single high precision multiplier or as a set of lower precision multipliers.

The operation of the recursive multiplier starts with the simultaneous execution of $(N/q)^2$ partial multiplications using $(N/q)^2$ P_q-multipliers.

A P_q-multiplier is a device that performs the following operation:

$$O(m,n) = \left(\sum_{i=0}^{q-1} h_{m+i} 2^{m+i}\right)\left(\sum_{j=0}^{q-1} k_{n+j} 2^{n+j}\right) \qquad (7)$$

In order to reduce the delay of the structure, O(m,n) is obtained as a sum of three numbers:

$$O(m,n) = \sum_{r=0}^{q-1} o_r 2^{m+n+r} + \sum_{r=q}^{2q-1} a_r 2^{m+n+r} + \sum_{r=q}^{2q-1} b_r 2^{m+n+r}$$

$$O(m,n) = L_1(m,n) + M_1(m,n) + M_2(m,n) \qquad (8)$$

Fig. 1a showns the layout of a P_q-multiplier using the cells whose design will be introduced in [11]. The area complexity of this multiplier is

$$A_q = \left(\frac{q}{2}\right)^2$$

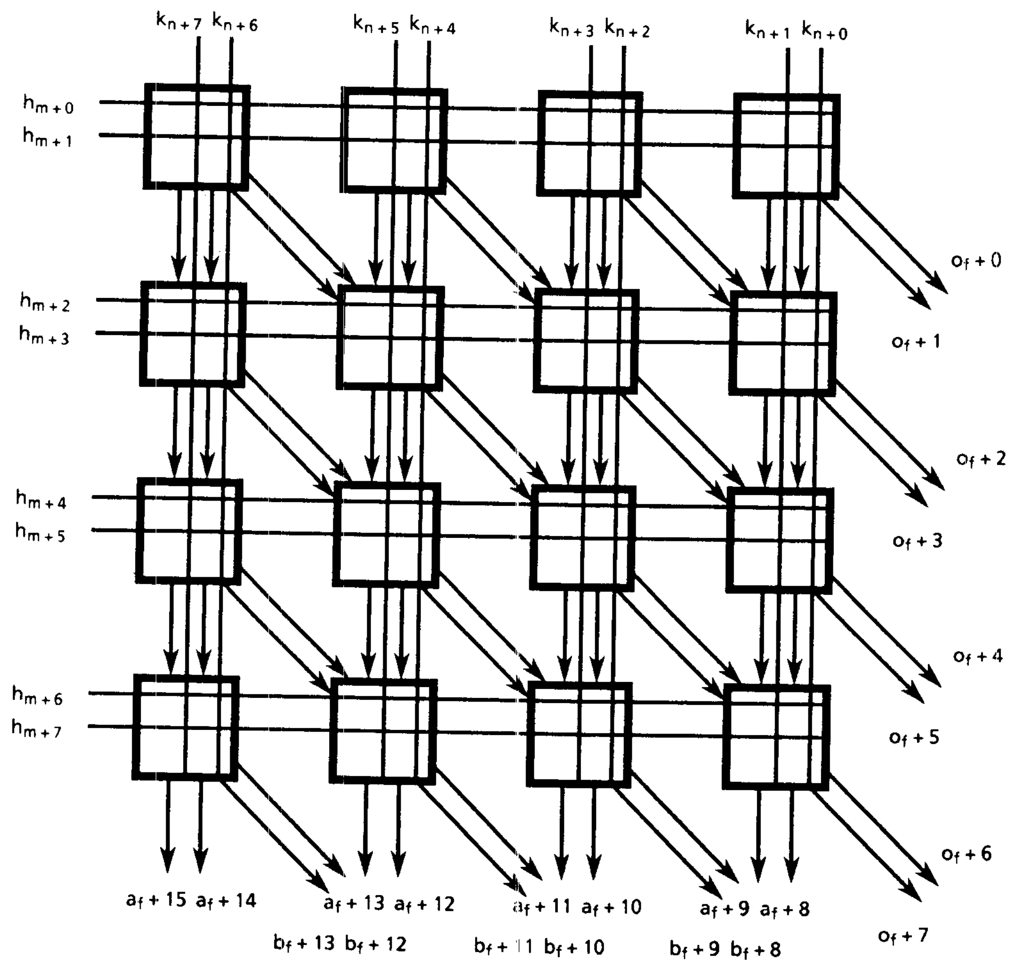
and the time complexity is

$$T_q = qT_m$$

Fig. 1b shows a schematic representation of a P_q-multiplier.

Each macro cell in Fig. 1a is, a ***full multiplier*** with G = 2.

The output of the P_q-multipliers have to be added together in order to give the final product. A first step towards the calculation of the final sum is that of grouping the P_q-multipliers into squares containning 4 P_q-multipliers each and to perform a pseudo-addition of the output of the 4 P_q-multipliers of each square.



$f = m + n$

FIG. 1a Iterative P_q-multiplier (detailed scheme)

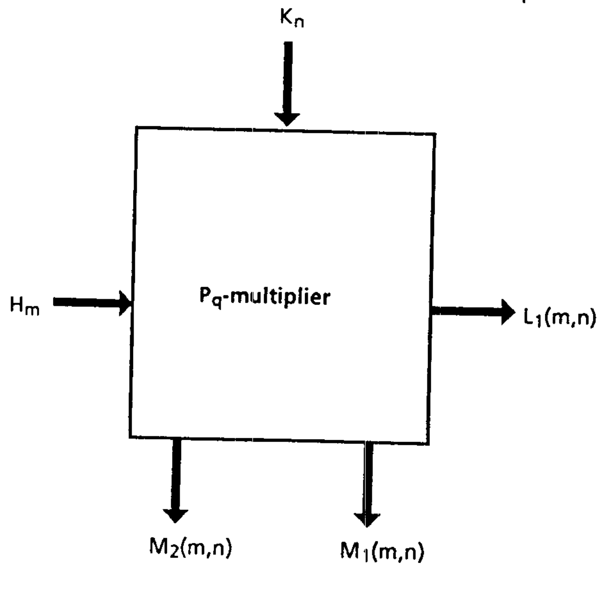


FIG. 1b Iterative P_q-multiplier

Fig. 2 shows how the output of four P_q-multipliers can be added toghether using carry-save adders.

Let us call the structure of fig 2 a ***B_1-pseudomultiplier***.

The output $O_1(m,n)$ of the B_1-pseudomultiplier can be expressed as follows:

$$O_1(m,n) = O_{11}(m,n) + O_{12}(m,n) \tag{9}$$

where

$$O_{11}(m,n) = OL_{11}(m,n) + OL_{21}(m,n) + OM_{11}(m,n) + OM_{21}(m,n)$$

$$O_{12}(m,n) = OL_{22}(m,n) + OM_{12}(m,n) + OM_{22}(m,n)$$

$$OL_{11}(m,n) = L_1(m,n)$$

$$OL_{21}(m,n) + OL_{22}(m,n) = M_1(m,n) + M_2(m,n) + L_1(m,n+q)$$

$$OM_{11}(m,n) + OM_{12}(m,n) = L_1(m+q,n+q) + M_1(m,n+q) +$$

$$M_2(m,n+q) + M_1(m+q,n) + M_2(m+q,n)$$

$$OM_{21}(m,n) = M_2(m+q,n+q)$$

$$OM_{22}(m,n) = M_2(m+q,n+q)$$

The B_1-pseudomultipliers are grouped into sets of four each. The outputs of each set are added by two carry-save adders using the same structure as in Fig. 2 but with six inputs for each weight.

Let us call the structure of 4 B_1-pseudomultipliers a ***B_2*-*pseudomultiplier***.

We continue in this way until we introduce a B_p-pseudomultiplier that contains all the P_q-multipliers.

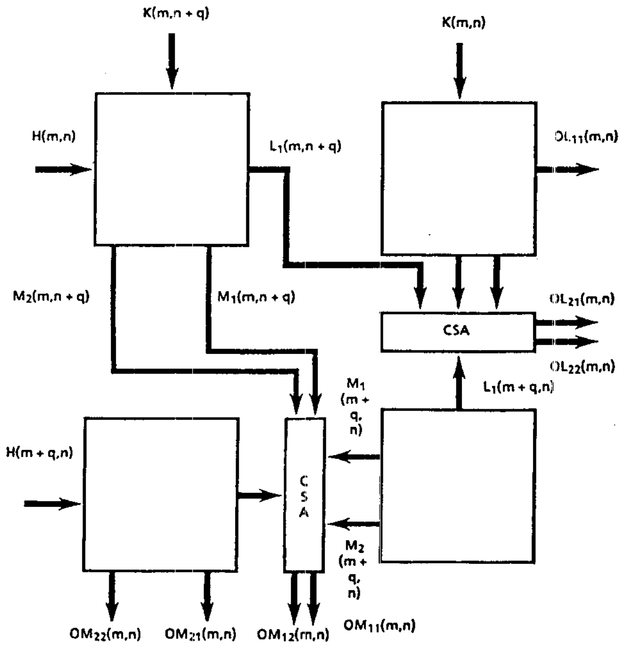


FIG.2 B_1-pseudo-multiplier

ALGORITHM RM (Recursive Multiplier)

The B_p-pseudomultiplier outputs two numbers Op_1 and Op_2 that have to be added by a special adder.

The algorithm of the B_p-multiplier can be descrided as follows:

```
Bp-multiplier(n,H,K,Op1,Op2);
  input :H,K,n;
  output:Op1Op2;
begin
  If n = N/q then
    Pq-multiplier(H,K,Op1,Op2);
  else
    cobegin
      Bp-multiplier(H[0,0],K[0,0],Op1[0,0],Op2[0,0]);
      Bp-multiplier(H[n/2,0],K[n/2,0],Op1[n/2,0],Op2[n/2,0]);
      Bp-multiplier(H[0,n/2],K[0,n/2],Op1[0,n/2],Op2[0,n/2]);
      Bp-multiplier(H[n/2,n/2],K[n/2,n/2],
                    Op1[n/2,n/2],Op2[n/2,n/2]);
    coend;
end;
```

The algorithm of the entire multiplier can be described as follows:

```
multiplier(H,K);
inputs :H,K;
output :multiplier;
begin
  Bp-multiplier(N,H,K,Op1,Op2);
  multiplier: = special-adder(Op1,Op2);
end;
```

The multiplication is executed by the recursive algorithm **B_f-multiplier** ($1 \le f \le p$).

Fig. 3 showns a schematic layout for a B_1-multiplier, connection between macrocells and networks of CSA have been omitted for the sake of simplicity.

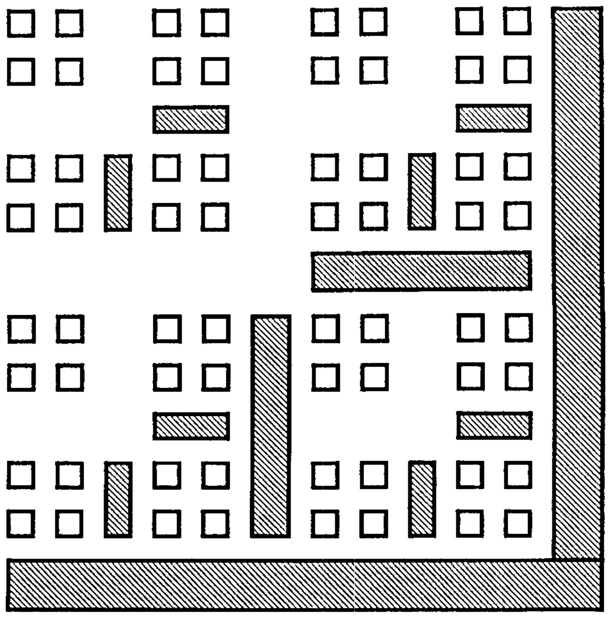


Fig. 3 Recursive multiplier layout

Fig. 4 showns an example of the multiplication executed by a recursive multiplier using positive numbers represented in the decimal code.

The pseudo-adders are represented in Fig. 3 by dashed lines. The maximum number of input addends is 6 and wires carrying pairs of addends are already ordered in such a way that wires carrying bits of the same weight are adjacent.

The layout of a pseudo-adder is sketched in Fig 5. Capital letters indicate addends and lower case letters indicate bits.

The time complexity of the structure is O(3).

The area complexity of the pseudo-adder of all the B_f-multiplier is $O(\alpha N/2)$. Where α is the ratio between the area complexity of a pair of wires and the area of a P_q-multiplier divided by the number of bits of the P_q-multiplier.

TIME COMPLEXITY

Assuming that D is the delay introduced by an AND/OR expression implementing the basic functions of a CSA and the

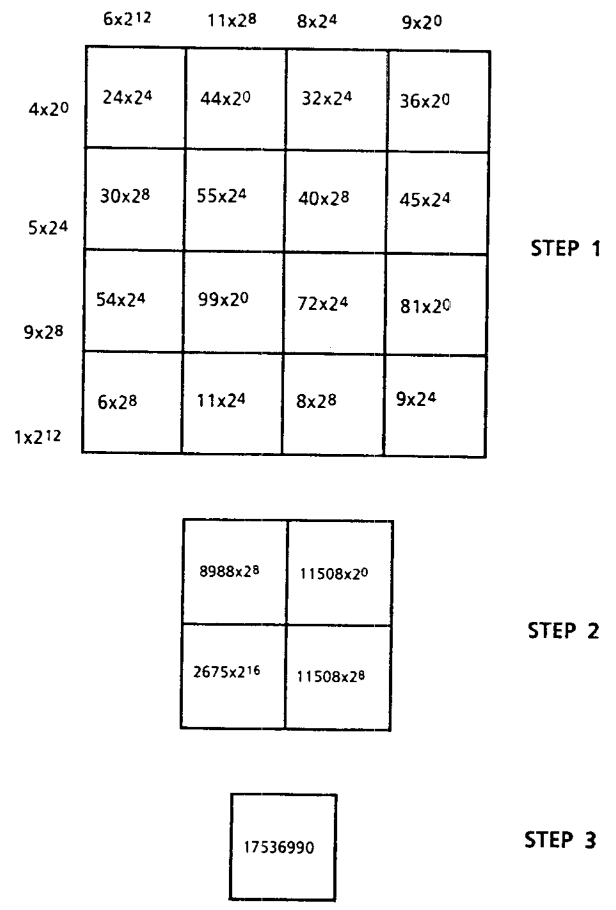


FIG.4 Recursive multiplication example

Special Adder (SA), assuming T(q) to be the delay introduced by a P_q-multiplier, if p recursions are applied, then the total delay can be expressed as follows:

$$T = (log2N + 3p)D + T(q) \qquad (10)$$

Using the just described desing aproach, a cellular p_q-multiplier of N/q-bit can be design with a delay:

$$T(q) = (\frac{N}{2q})D$$

The P_q-multiplier could be also implemented with a Read Only Memory (ROM) making T(q) independant of N. In this case the (10) can be rewritten as :

$$T = (log2N + 3p) + T(ROM)$$

A technologically acceptable solution could be a ROM with a number of bits less that 2^{12}.

In order to find a relation between p and q, notice that there are q^2 P_q-multipliers that are grouped into group of 4 giving $q^2/4$ B_1-multipliers. B_1-multiplier are grouped into groups of 4 giving $q^2/4^2$ B_2-multipliers and so on until 1 B_p-multiplier is obtained. Thus:

$$p = \lceil log_2 q \rceil$$

Where $\lceil A \rceil$ means A if A is integer or the least integer greater than A.

Notice that

$$q < N; \quad p < log N$$

In order to mantain logarithmic the time complexity of the structure, several condition on p can be imposed. A very simple one is the following:

$$\frac{N}{q} = log\, N; \quad q = \frac{N}{log\, N} \qquad (11)$$

AREA COMPLEXITY

The area complexity of the recursive multiplier can be computed by inspection of Fig. 3 as follows:

$$A = O(A(B_2 - multiplier)) + O(Nlog\, N)$$

$$= O(4A(B_1 - multiplier)) +$$

$$+ O(NlogN) + area\ of 2\frac{N}{2}\ pseudo-adders$$

$$= O(16A(P_p - multiplier)) +$$

$$area\ of 2\frac{N}{2} pseudo-adder +$$

$$area\ of\ 4\frac{N}{4}\ pseudo-adder + O(N, logN)$$

The area complexity of q^2 P_q-multipliers is $O(q^2(N/q)^2) = O(N^2)$.

There are p = log q ≈ log(N/logN) levels of pseudo adders. Their overall area complexity is:

$$A = O(\alpha(2\frac{N^2}{4} + 8\frac{N^2}{16} + ...)) = O((\frac{\alpha}{2})N^2 p) \qquad (12)$$

The overall area complexity is:

$$A = O(N^2(1 + \frac{\alpha}{2} log\, q) + O(N\, log\, N) \qquad (13)$$

Thus the overall area complexity is $O(N^2)$ as far as $(\alpha/2)logq < 1$.

Given the proposed structure and the cell design proposed in [11], α is the ratio between the area complexity of two wires and the area complexity of a 16 input multiplexers plus 20 circuits implementing boolean functions of 4 variables.

An acceptable estimation could be α = 1/20.

The area complexity of the proposed structure can be assumed to be $O(N^2)$ as far as

$$log(\frac{N}{log\, N}) \leq 40 \qquad (14)$$

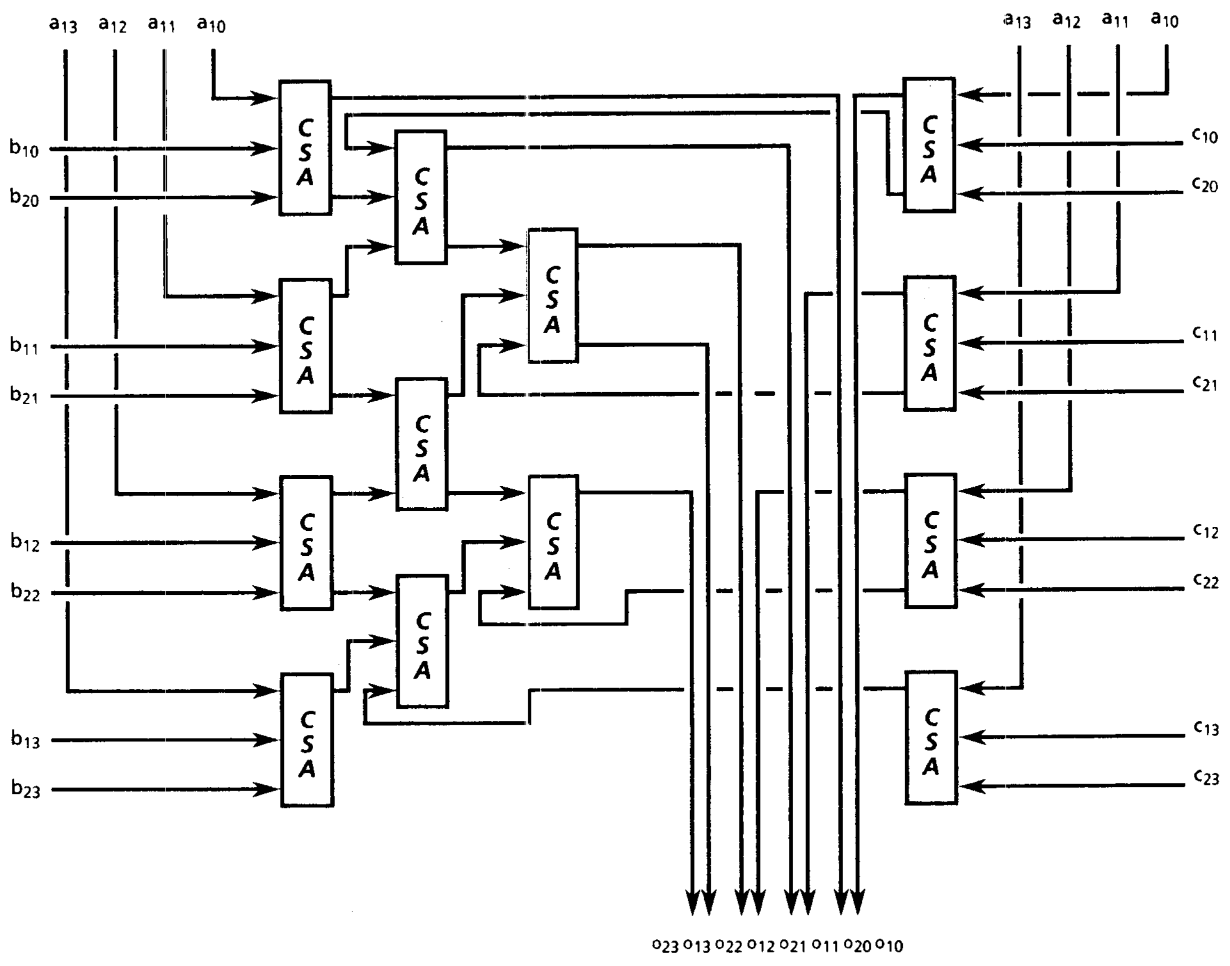
which is an acceptable condition for a large class of practical multipliers.

VLSI COMPLEXITY

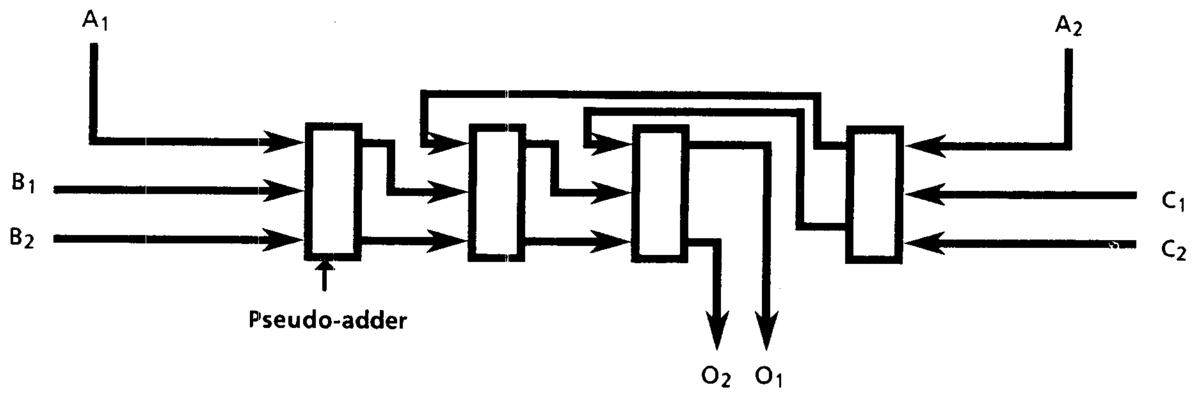
A relation is currently used for measuring the **VLSI** complexity of a proposed structure [3,5]:

$$F = AP^2T^2$$

where, A is the complexity, P is the period complexity and T is the



(detailed scheme)



(Global scheme)

Fig. 5 Pseudo-adder scheme

latency complexity. We will first assume that the structure is not pipelined, for the sake of simplicity. The latency complexity T is O(logN) and the area complexity A is $O(N^2)$.

CONCLUSION

A network for performing multiplication of two binary numbers has been proposed. The network can be implemented in a synchronous or in an asynchronous way. If the factors to be multiplied have N bits, the area complexity of the networks is $O(N^2)$ for practical values of N as in the case of cellular multipliers. The time complexity is O(log N).

With some additional circuits the multipliers can be used either independently for performing four separate single precision multiplications or connected to perform a single double precision multiplication.

These details as well as pipelining and making the cell capable of multiplying two two's complement numbers are omitted for the sake of brevity.

The proposed recursive scheme for parallel multiplication combines the advantages in area complexity of cellular multipliers with the advantages in time complexity of tree multipliers and may hopefully, suggest new design ideas for other types of units.

ACKNOWLEDGMENTS

This work was supported by the ***Natural Sciences and Engineering Research Council of Canada*** and the ***FCAC***.

REFERENCES

[1] Bandyopadhay, S., Basu, S. and Choudhury, A.K., ***An iterative array for multiplication of signed binary numbers.*** IEEE Trans. Comp., vol. C-21, pp. 921-922, Aug 1972.

[2] Baugh, C.R. and Wooley, B.A., ***A two's complement parallel array multiplication algorithm,*** IEEE trans. Comp., vol C-22,pp. 1045-1047, Dec 1973.

[3] Brent,R.P. and Kung, H.T., ***The chip complexity of binary arithmetic,*** In proc. 12 Annual ACM symp. Theory of computing, Los Angeles, Cal., April 28-30, 1980, ACM, new York, pp. 190-200.

[4] Burton, D.P., Byrne, P.C. and Noaks, D.R., ***A multiplier for complex binary numbers,*** Electronic Eng., pp. 71-73, April 1970.

[5] Cappello, P.R. and Steiglitz, K., ***A VLSI layout for a pipelined Dadda multiplier***, ACM Trans. Comp. Sys., vol.1, May 1983.

[6] Dadda, L. ***Some schemes for parallel multipliers,*** Alta frequenza, pp. 349-356, May 1965. Reprinted in Computer Design Developpement, E.E. Swartzlander, Jr., Ed. Hayden book, Rochelle Park, N.J., 1976.

[7] Dean, K.J., ***Cellular multiplier subarrays: a critical-path approach to propagation time,*** Electronics Lett., vol. 7, pp. 75-77, Feb 1971.

[8] Deegan, I.D., ***Cellular multiplier for signed binary numbers,*** Electronics Lett., vol. 7, pp. 436-437, July 1971.

[9] De Mori, R., ***Suggestion for an IC fast parallel multiplier,*** Electronics Lett., vol. 5, pp. 50-51, Jan 1969.

[10] De Mori, R. and Serra, A., ***A parallel structure for signed-number multiplication and addition,*** IEEE Trans. Comp., vol. C-21, pp. 1453-1454, Dec 1972.

[11] De Mori,R. and Cardin,R. ***A parallel multipler based on multiplexers*** Signal processing, vol 6, pp. 213-223, june 1984.

[12] De Mori, R., Riviora, S. and Serra, A., ***A special-purpose computers for digital processing,*** IEEE trans. Comp., vol. C-24, pp. 1202-1211, Dec 1975.

[13] Fletcher, W.I., ***An Engineering approach to digital design,*** Englewood Cliffs, Prentice-Hall, 1980.

[14] Guild, H.H., ***Fully iterative fast array for binary multiplication and fast addition,*** Electronics Lett., vol 5, pp. 263, May 1969.

[15] Habibi, A. and Wintz, P.A., ***Fast multipliers,*** IEEE Trans. Comp., vol C-19, pp. 153-157, Feb 1970.

[16] Hoffman, J.C., Lacaze, B. and Csillag, p., ***Multiplieur parallele a circuit logiques iteratifs,*** Electronics Lett., vol. 4, pp. 178-179, May 1968.

[17] Kingsbury, N.G., ***High Speed binary multiplier,*** Electronics Lett., vol. 7, pp. 277-278, May 1971.

[18] Hwang, K., ***Computer arithmetic: Principales, Architecture and Design,*** New York, Wiley, 1979.

[19] Ling, H., ***High speed computer multiplication using a multiple-bit decoding algorithm,*** IEEE Trans. Comp., vol C-19, pp. 706-709, Aug 1970.

[20] Majithia, J.C. and Kitai, R., ***An iterative array for multiplication of signed binary numbers,*** IEEE Trans. Comp., vol. C-20, pp. 214-216, Feb 1971.

[21] Preparata, F.P., ***A mesh-connected Area-time Optimal VLSI multiplier of large integers,*** IEEE Trans. Comp., vol c-32, pp. 194-198, Feb 1983.

[22] Singh, S. and Waxman, R., ***Multiple operand addition and multiplication,*** IEEE Trans. Comp., vol C-22, pp. 113-120, Feb 1973.

[23] Springer, J. and Alfke, P., ***Parallel multiplier gets boost from IC iterative logic,*** Electronics, vol. 43, pp. 89-93, Oct 1970.

[24] Stenzel, W.J., Kubitz, W.J. and Garcia, G.H., ***A compact high-speed parallel multiplication scheme,*** IEEE trans. Comp., vol C-26, pp. 948-957, Oct 1977.

[25] TRW, LSI products, **LSI multipliers**, Redondo Beach, CA, Aug 1978.

[26] Wallace, C.S., ***A suggestion for a fast multiplier,*** IEEE Trans. Electron. Comp., pp. 14-17, Feb 1964.