Reprinted from *Alta Freq.* 45:574-580 (1976)

ON PARALLEL DIGITAL MULTIPLIERS

L. DADDA (*)

I. - Introduction.

Parallel multipliers consist of a combinatorial network capable of obtaining the product of two binary numbers. Parallel multiplier are to be compared with multipliers used by the well-known process of repeated additions of the multiplicand according to the value of the multiplier's bits (1).

The complexity and the cost of parallel multipliers is larger than the one of standard multipliers: what is sought with parallel multipliers is speed. It can be shown that for several applications the overall increase in computation speed largely compensates the additional cost of a parallel multiplier. Moreover, a fast multiplier is essential in some application, such as real time signal processing.

Several schemes for parallel multipliers have been proposed [2 ÷ 6]. They can be ranged in two classes: in the first class [4, 5, 6] a « cellular » structure (i.e. a rectangular array of identical cells) is used; in the second calss [2, 3] a « reduction » scheme is used, where the set of summands (each consisting of the multiplicand logically multiplied by the multiplier bits and suitable shifted) are reduced to two numbers, whose sum equals the product. A comparison among some schemes has been given in [7, 8].

It appears that, although cellular structured multipliers are appealing as far as uniformity of components is concerned, they are slower than multipliers based on the reduction schemes, which minimize the number of logical levels.

The purpose of this paper is to show how the scheme proposed in [3] (which is of the « reduction » type) can lead to the implementation of parallel, fast multipliers for numbers of up to several tens of bits (e.g. 30 ÷ 40) with a total delay of about 100 ns, using today's components and at a reasonable cost. Suggestions shall also be given for new components in order to obtain a cost reduction and higher speed.

II. - Parallel multipliers: « reduction » schemes.

In this chapter we summarize the main concepts presented in [3] with some additional comments.

The basic scheme is given in fig. 1, where, in the upper array of dots, each row represents the multiplicand, logically multiplied by one of the bit of the multiplier (the least significant bit of the multiplicand is at the right, the least significant bit of the multiplier is at the top).

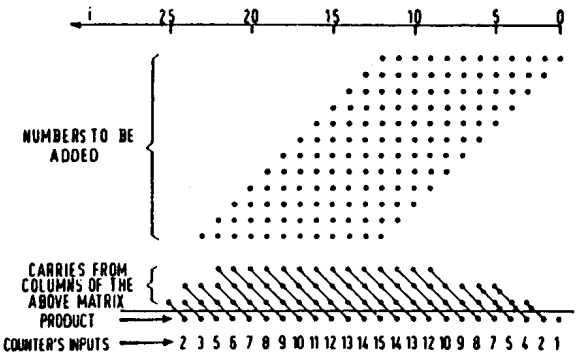

Fig. 1. - Multiplication (12 × 12 bit) through addition, in a single stage, using a parallel counter for each column. Carries are propagated through the counters.

The product to be obtained is the binary sum of all rows (summands set). Such a sum can be obtained using a « parallel counter » for each column. A parallel (p, q) counter, see fig. 2, is a combinatorial network with q outputs and $p \leqslant 2^q - 1$ inputs, where the binary number represented by the q outputs is the number of « ones » present at the inputs.

Fig. 2 gives also the « shorthand » representation of parallel counters, as it will be used in all remaining figures of this paper, and it shows that full adders and half adder can be considered as (3, 2) and (2, 2) counter, respectively.

The input to the ith counter (for the ith column) is given by the bits in the ith column of the summands set, and also by the outputs (carries) from the lower order counters.

Although the scheme in fig. 1 is the simplest from the conceptual point of view, it has some implementation drawbacks. First of all, parallel counters for a larger number of inputs (fig. 1 reguires parallel (15,4) counters) are difficult to built. Moreover, they have a large delay, which contributes to the total multiplication time through the carries propagation along the chain of parallel counters.

In order to minimize the carries propagation delay, the process can be dividend into two steps. In the first step, from the original set of summands a set of two numbers is obtained, whose sum equals the

(*) Luigi Dadda (Socio AEI) - Politecnico di Milano, Istituto di Elettrotecnica ed Elettronica, Centro di studio per l'Ingegneria dei sistemi di elaborazione delle informazioni, del Consiglio Nazionale delle Ricerche.

(1) In such multipliers, the effective number of additions can be made a fraction of the number of the multiplier bits, by special arrangements, such as suitable coding of the multiplier [1].

product. In the second step the product is obtained by adding this two numbers.

The reason for the two-step process is that the first step can be accomplished using a network with a small number of logical levels, and correspondingly with a small total delay, as no carries propagation is allowed in it; carries propagation is confined in the second step, where it can be accomplished in a very effective manner using fast adder based on look-ahead arrangements.

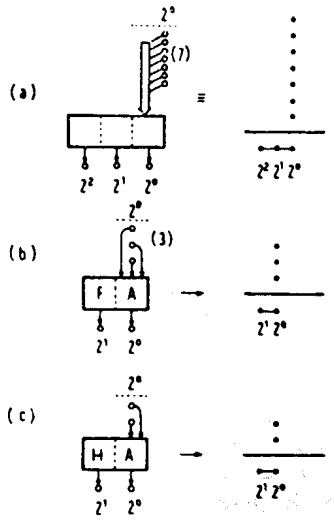

Fig. 2. - « Simple » parallel counters: *a*) a 7 inputs-3 outputs, or (7,3) parallel counter and its « shorthand » notation used in the following figures; *b*) a (3,2) parallel counter, i.e. a full adder; *c*) a (2,2) parallel counter, i.e. a half adder.

A second scheme, based on such principle, and still using parallel counters (with no limitation on the number of their inputs) is given in fig. 3. The number of logical levels (vs. the number of bits of the multiplier) required for such a multiplier is given in fig. 4 ([2]).

Since parallel multipliers with a large inputs number are difficult to built (and are in general slower than parallel counters with a limited number of inputs), a final step in the development of a feasable parallel multiplier based on the above concept, is to prescribe parallel counters with a limited number of inputs.

It is shown in [3] how parallel multipliers can be built using (2,2) and (3,2) parallel counter, i.e. half and full adders.

Fig. 5 gives an example of such a multiplier, and fig. 4 gives the number of stages required (vs. the number of bits of the multiplier ([3]).

It has to be noted that the Wallace scheme [2] for parallel multipliers belongs to the above category.

In restricting the counter's inputs number, the total number of logical levels and the number of counters, necessary for the implementation of a multiplier, increases.

([2]) A question might arise, in connection with the results given in fig. 4, whether a smaller number of logical level could be obtained using parallel counters of more complex design. The question is treated in chapter IV.

([3]) A 12 × 12 multiplier of such scheme was built [9] using full adders based on threshold logic circuits and parallel adder with a fast (1 ns/stage) carry propagati- tal multiplication time of 550 ns.

It is therefore important, both from the point of view of cost and speed, to look for parallel counters with a large number of inputs taking care of the fact that counters with a large number of inputs can be both too expensive and slow.

We shall not try to give an exact solution to such an optimization problem: we wish only to suggest that if in the past the only feasible reduction multiplier had to be based on a network of full and half adders (being the only integrated, low cost parallel counters available) the technological progress offers nowadays the possibility to use parallel counters with a larger number of inputs, e.g. 7 inputs, and even more.

We want therefore to show in the next chapter how these counters can be built, particularly (7,3) counter which seems (see next chapter) at the present state of I.C. technology a good solution both from speed and cost view-points.

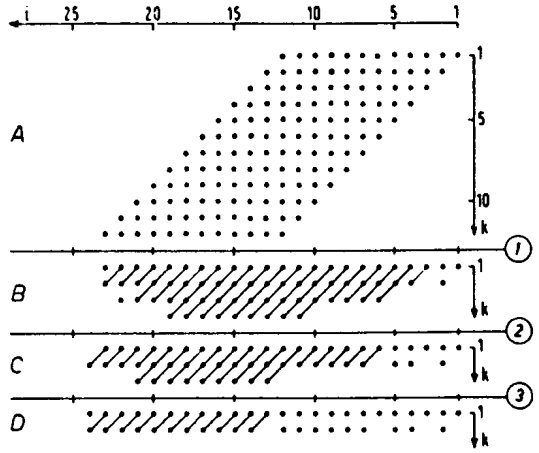

Fig. 3. - A multiplier scheme, obtaining two numbers whose sum equals the product, of 12 x 12 bit, through three stages, using a parallel counter for each stage and for each column. Carries are propagated only in the addition of the two final numbers.

In chapter IV some special kind of counters, and their use in parallel multipliers, shall be illustrated.

Fig. 6 shows as an example the scheme of a (15 × 15) parallel multiplier using (7,3) counters (along with (3,2), (2,2) counters when necessary).

Fig. 4 gives the numbers of levels (vs. the number of bits of the multiplier) for multipliers using (7,3) and (15,4) parallel counters.

The above schemes and fig. 4 are drawn according to procedures illustrated in [3].

III. - The feasibility of parallel counters for a large number of inputs.

It shall be now shown how parallel counters, can be conveniently implemented using available LSI components ([4]).

Such components are ROM (PROM) memories and/

([4])When this paper was already completed, the reference [8] came to our attention. This report discusses the design of parallel multipliers based on the reduction scheme and using parallel counters implemented with ROMs or PLAs, as suggested in this paper. As the Stenzel's approach stresses the use of non-simple counters (particularly (5,5; 4) counters) whereas we emphasize the use of simple counters, the content of the two papers is largely non overlapping and complementary.

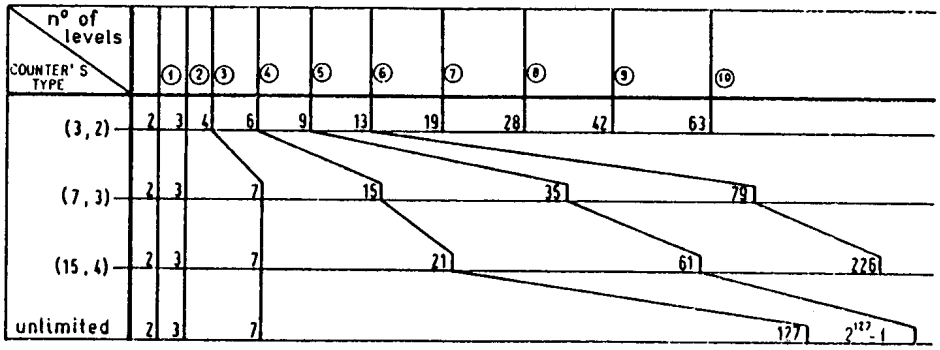

Fig. 4. - Number of levels (vs. number of multiplier's bits and parallel counter's type) in the reduction network (e.g. for $10 \div 13$ bit multipliers, 5 levels are required using (3,2) counters; for $8 \div 15$ bit multipliers, 3 levels are required using (7,3) counters; for $8 \div 127$ bits multipliers, 3 levels are required using counters with unlimited number of inputs).

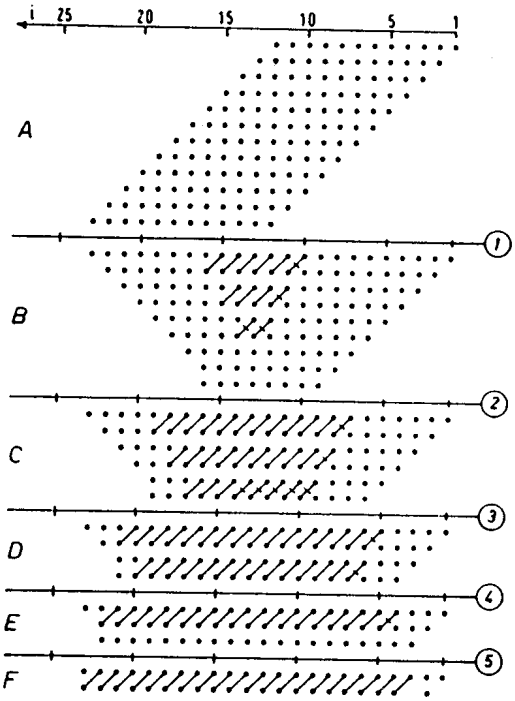

Fig. 5. - The reduction network of a 12×12 bit multiplier (i.e. a network obtaining two numbers whose sum equals the product) using (3,2) and (2,2) counters.

or PLA (Programmable Logical Arrays). ROM memories shall be examined in the following with some detail (5).

It is first to be noted that PROMs of the fusible link type (i.e. PROMs where the content of each cell can be « written » by blowing a fuse by means of current pulses) are commercially available with a capacity of 256×4 and with a typical access time of 25 ns, using Schottky TTL logic. Faster PROMs of the same capacity using ECL logic and with smaller access time (15 ns) are also available.

In order to obtain (7,3) counters it is only necessary to « program » the content of such PROMs in such a way that each of the $2^7 = 128$ words are written with the binary number equal to the number of « ones » in the corresponding 7-bit address. Note that, in such a way, only $128 \times 3 = 384$ cells (over 1024), and correspondingly only 7 address (input) pins (over 8) and 3 output pins (over 4) shall be used.

This suggests that, in order to decrease the component count for a given multiplier, special I.C.s, designed for use in parallel multipliers, could be produced using the same technologies used in today's PROMs and having the same degree of complexity or chip area.

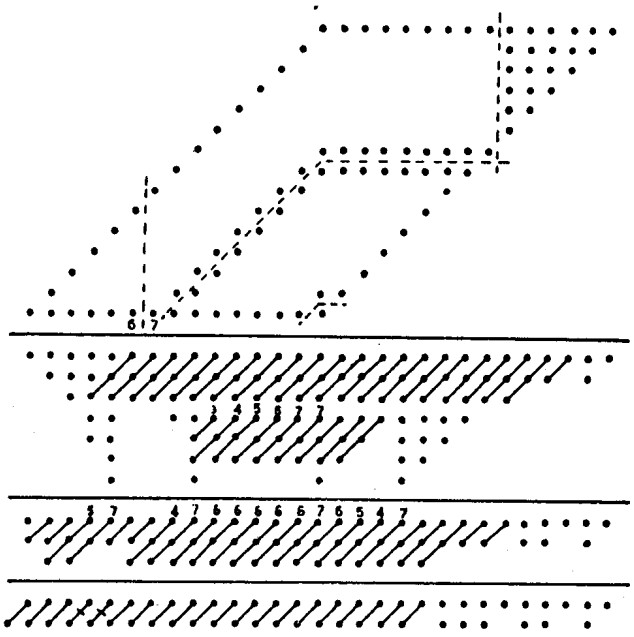

Fig. 6. - The reduction network of a 15×15 bit multiplier using (7,3), (3,2) and (2,2) counters.

The first suggestion is to implement two (7,3) counters on the same chip and using a package having 2×7 inputs and 2×3 outputs (e.g., a 22 pins standard dual in line package).

The second suggestion deals with the problem of obtaining the initial summands set. In the preceding

(5) Though a full investigation has not been made, it appears that ROMs are preferable to PLAs, due to the large number of implicants involved in the two-level implementation of the functions describing a counter.

Available PLAs have in effect a relatively small number of realizable products, whereas with ROM all minterms are implemented. plemented.

Moreover, the speed of available PLAs is smaller than the speed of the ROMs mentioned in the text.

Parallel counters with more than three inputs can also be obtained by composing several full adders, using the same rules illustrated in [3]. In fig. 7 a (7,3) counter's scheme is shown: it requires four full adders and three reduction levels.

Such a scheme has been implemented in integrated form by Texas Instr. (SN 54/74S275, with a typical delay of 45 ns).

schemes, this is assumed as obtained through an AND gates for each element.

Though conceptually simple, this solution requires a large number of AND gates and a correspondingly large number of I.C. packages. Although such components are of the cheapest type, the cost of obtaining the summands set is a relatively large part of the total multiplier cost, due to the wiring, and printed circuit board area.

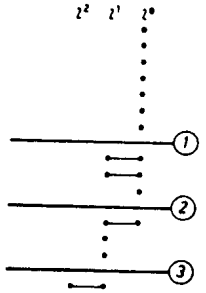

Fig. 7. - A (7,3) parallel counter, assembled from four (3,2) counters.

Most of the AND gates could conveniently be implemented on the same chips implementing (7,3) counters, with a small increase in chip-area.

A single (7,3) counter with the 7 corresponding AND gates could be housed in a package with $2 \times 7 = 14$ input pins and 3 output pins (fig. 8 *a*).

As far as the chip-area is concerned, two (7,3) counter could also be accomodated on the same chip: a difficulty arises in relation to the number of pins, if two totally independent counter are considered (i.e., with independent inputs), since a package with a very large number of pins would be necessary.

In order to avoid this difficulty, we can notice that many of the (7,3) counters as used in the preceding schemes can be paired, as shown in fig. 8 *b* and *c*, in such a way that their inputs can be generated using 7 multipliers bits and only 8 multiplicand bits.

The fig. 8 *b* scheme applies well in the left-hand part of the summands set, whereas in the right-hand part it is useful to have an input pairing as shown in fig. 8 *c*. In order to avoid two distinct chips, a single chip could be designed, with 2×8 inputs, and an additional pin permitting to « program » the chip as in fig. 8 *b* or in fig. 8 *a*, according to the need.

The implementation of larger parallel counters following the above suggestions, doesn't seem advisable, due to the size of the chip and the delay. For instance, the (15,4) parallel counter (which

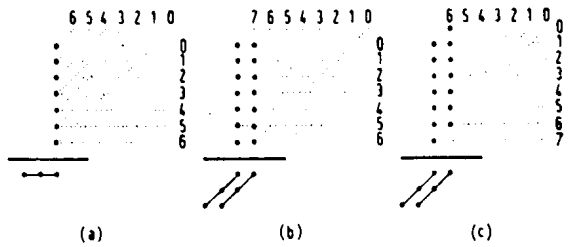

Fig. 8. - Schemes of single (*a*) and paired (*b*, *c*) (7,3) counters, to be used in the first stage of the reduction network, having as inputs the multiplicand's bits (represented by the diagonal dotted lines) and the multiplier's bits (represented by the horizontal lines).

needs 15 inputs and 4 output pins) could be realized by means of a ROM, requiring a chip of $2^{15} \times 4 = 32\,k \times 4 = 128$ k bits, which is unfeasible with present technologies.

A scheme for obtaining a (15,4) counter with an acceptable chip size could use two (7,3) counters, whose outputs are added in a parallel 3 bits adder (with an initial carry to take into account the 15th bit to be counted.

Although this scheme seems feasible with todays technologies, it is to be noticed that its delay shall be somewhat larger than for a (7,3) counter.

Another more effective solution of the problem (of including the initial summands generation in the first reduction stage) can be achieved through sub-multipliers for a small number (e.g. 4×4) of bits, where each product's bit is obtained as a direct function of the factor's bits.

The use of sub-multipliers has been thoroughly studied by Ferrari and Stefanelli [11] (see also [8]) (6).

Fig. 9 shows the scheme of a 12×12 bits multiplier using (4×4) sub-multipliers.

The main advantage in using sub-multipliers in the first stage is the reduction of the overall number of pins in the reduction network. Fig. 9 uses a total of 300 pins, whereas the generation of the $12 \times 12 = 144$ initial summands' bits would require 3×144 pins if implemented by means of individual 2 inputs AND gates.

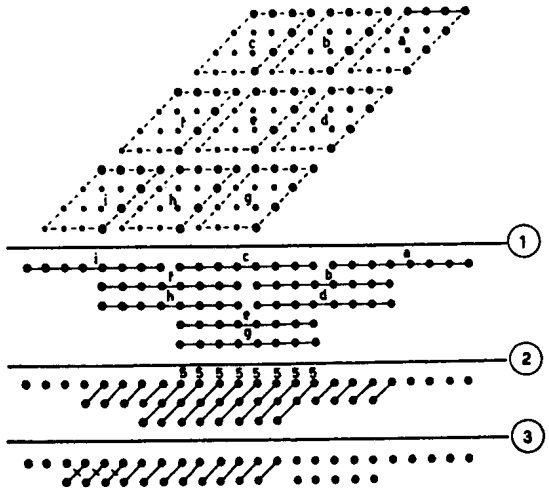

Fig. 9. - The reduction network of a (12×12) bit multipliers using (4×4) sub-multipliers in the first stage.

As already said, to obtain a complete multiplier we need to sum the two numbers obtained at the outputs of the reduction network composed of parallel counters. To obtain a fast adder we can use I.C. 4-bit adders or ALUs, with carry-look-ahead units in order to minimize the carry propagation delay. Using available components we can implement an adder for 32 bits (to be used for a 16×16 bits multiplier) with a total addition of about 40 ns, or an adder for 64 bits (for a 32×32 bit multipliers) with a total addition time of about 50 ns.

Taking into account the number of logical levels in the reduction networks (fig. 4) and the typical

(6) Such sub-multipliers are offered by some I.C. manufacturers. They can also be obtained by me[illegible]s.

delay of (7,3) counters as described (15 to 25 ns) we can state the feasibility of a parallel multiplier with about 100 ns for numbers lengths of 30 ÷ 40 bits.

It is interesting to note [3] that a more carefull evaluation of the total multiplication time has to take into consideration the fact that the effective number of levels in the reduction network is increasing in steps from zero to the maximum (given by fig. 4), proceeding from the least significant bits. This means that the total multiplication time is certainly smaller than the sum of the carry propagation time in the adder and the maximum delay in the reduction network.

IV. - Parallel counters of more general type and their use in the reduction process.

Parallel counters as defined in chapter II (fig. 2) can be considered as a particular case of a more general class af arithmetic elements defined by Meo [10].

Whereas parallel counters as in fig. 2 have all inputs having 2^0 weight, such elements can have also inputs having larger weights, i.e., 2^1, 2^2, etc., see fig. 10.

Such elements can still be considered as counters: we shall call them as « non-simple » ith-order parallel counters when 2^i is the highest input weight. « Simple » « 0-order » counters are the one defined in part II.

A non-simple counter can be represented by the notation: $(p_i, p_{i-1}, \ldots, p_0; q)$ where p_i represents the number of i-th order inputs (whose weight is 2^i), and q is the numer of outputs.

Its « capacity » C is:

$$C = \sum_0^i{}_k \; 2^k \cdot p_k$$

In general it will be:

$$C \leqslant 2^q - 1$$

A counter will be called « saturated » when

$$C = 2^q - 1$$

and « non saturated » otherwise. In a saturated counter all output combinations are significant, whereas in a non-saturated counter some output combinations shall never occur.

Such counters can be useful in some particular cases, as shall be shown in the following, for reducing the number of logical level in the summands set reduction process (fig. 9, 10, 11).

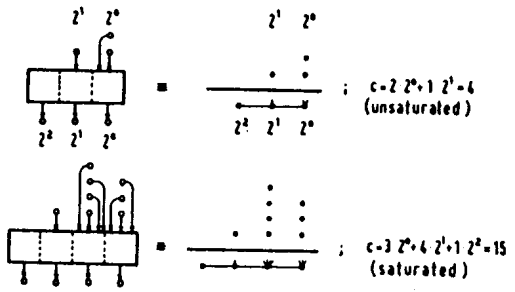

Fig. 10. - Generalized (« non-simple ») parallel counters: *a*) a (1,2; 3) unsatured counter; *b*) a (1,4,3; 4) saturated counter.

A particular class of such general type counter is represented by existing I.C. components performing $m \times n$ parallel multiplication for small m and n values (i.e., m, n = 2 or 4).

In fig. 10 *a*, a (1,2; 3) counter is represented. The number of outputs, 3, must accomodate the maximum value of the « capacity » (i.e. the sum of weighted inputs, all assumed as « 1 ») which, taking into account the different weights, is

$$C = 1 \cdot 2^1 + 2 \cdot 2^0 = 4$$

The same figure gives the short-hand notation of the (1, 2; 3) counter as used also in fig. 11; 12; 13.

Fig. 10 *b* represents a (1,4,3; 4) counter, whose capacity is $c = 1 \times 4 + 4 \times 2 + 3 \times 1 = 4 + 8 + 3 = 15$.

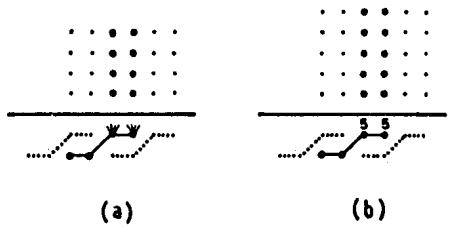

Fig. 11. - *a*) Reducing 4 rows to 2 rows using (4,4; 4) (unsaturated) counters; *b*) reducing 5 rows to 2 rows using (5,5; 4) (saturated) counters.

A first case of use of non-simple parallel counters, concerning the reduction process of the summands array, shall now be illustrated.

It happens in some cases that such reduction process using parallel counters reduces the original array to 4 rows: this requires a further step to obtain a 3 rows array before reaching the final 2 rows using a string of full adders (fig. 2, 5, 6).

We can ask whether it is possible to reduce from 4 rows to 2 rows directly, in a single logical level. This can be done as shown in fig. 11, where columns are paired and considered as inputs to a non-simple counter, having 4 outputs: two of these have weights 2^0, 2^1 (as the two paired input columns), the remaining outputs having weights 2^2, 2^3: they correspond to the low-weights outputs of the next counter, therefore forming a 2 row array.

The (4,4; 4) counter in fig. 11 *a* is non-saturated. The one in fig. 11 *b* is saturated ($2 \times 5 + 5 = 15$) and it allows the direct reduction from 5 rows to 2 rows.

The concept illustrated in the above examples can be generalized: its practical application finds nevertheless a limit in the size and complexity of the counters thus necessary.

In the case of fig. 11 *a* the (4,4; 4) counter can be implemented with a PROM, which uses a $2^8 \times 4 = 256 \times 4 = 1024$ bits ROM.

In the use of the above concept, consideration has to be given to the speed of the counters thus obtained, which is in general smaller than the speed of full adders.

A second case where non-simple counters can be used to decrease the number of logical levels in the reduction process, is illustrated in the fig. 12 example of a 16 × 16 multiplier.

In fig. 12 *a* simple (7,3), (3,2), (2,2) counters are used, and 4 stages appear necessary (see also fig. 4). Note also that in 15 × 15 case (fig. 6) 3 levels were necessary.

The fourth stage in the 16 × 16 case is due to the fact that after the first level a single column has 8 bits, and this requires three more stages.

Consider now fig. 12 *c*, and let us associate the two last bits of columns 16 and the last bit of the columns 15 at its left, and consider these three bits as inputs to a (1,2; 3) counter.

Using this counter in the first reduction stage, all the resulting columns have no more than 7 bits (fig. 12 *b*), so that only two more stages are necessary for the completion of the reduction.

In order to obtain the same result for a 17 × 17

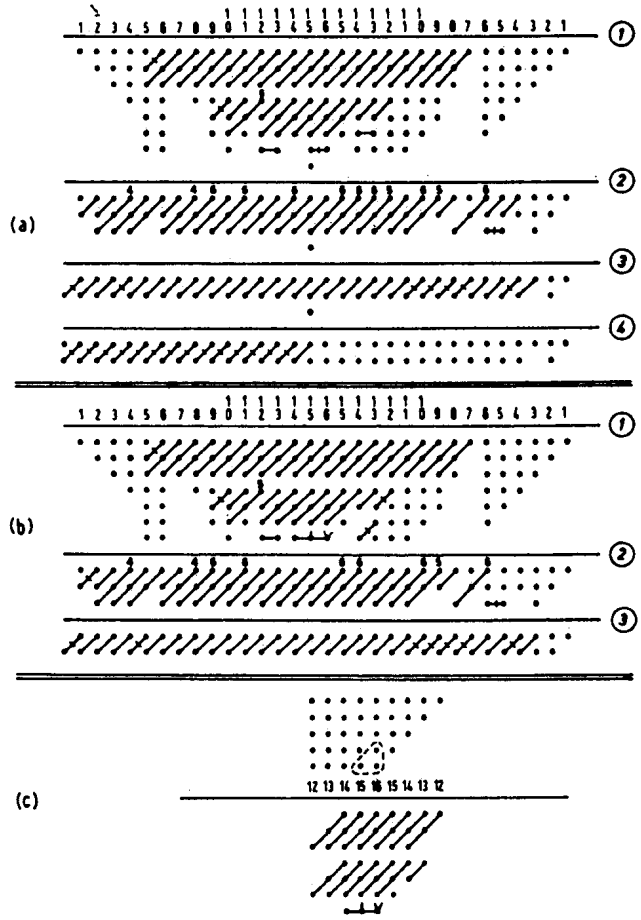

Fig. 12. - Decreasing the number of levels in the reduction network using non-simple counters. *a*) The reduction network for a 16 × 16 bits multiplier using only simple (7,3), (3,2), (2,2) counters; *b*) the reduction network using a single (1,2; 3) non-simple counter along with (7,3), (3,2), (2,2) counters; *c*) the dotted line shows the three bits in the summand array used as inputs to the (1,2; 3) non-simple counter.

multiplier (for which, see fig. 4, 4 stages are necessary using simple (7,3) counters) a more complex counter (1,2,3,2; 5) becomes necessary, as shown in fig. 13. Such a counter can be realized using a ROM having 7 inputs and 5 outputs, and requiring $2^7 \times 5 = 640$ bits.

Besides the above particular case, the use of non-simple counters doesn't seems in general convenient, for the following reason.

Considering the reduction step in the multiplication process (i.e. the reduction to two of the initial set of summands) it can be assumed that, in view of obtaining the minimum number of logical levels and of the total number of components (counters), it will be convenient to use counters whose ratio:

$$\rho = \frac{q}{p} = \frac{n^0 \text{ of outputs}}{n^0 \text{ of inputs}}$$

is as small as possible (ρ shall be colled « reduction factor »).

It can easily be shown that for a given q the minimum reduction factor is afforded by « simple » saturated counters.

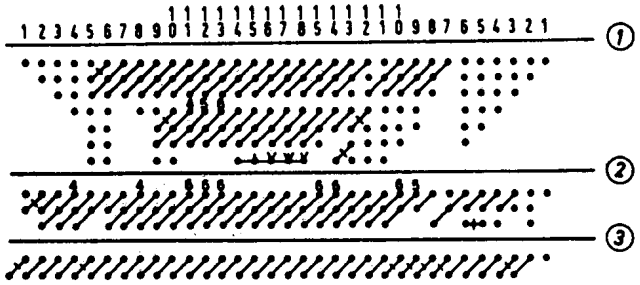

Fig. 13. - The reduction network of a 17 × 17 bit multiplier using a single (1,2,3,2; 5) non-simple counter, along with (7,3), (3,2) (2,2) counters, in order to decrease the number of levels.

In effect, if a « simple » counter is not saturated (for ex. a 6,3 counter), as it uses a number of inputs smaller than the maximum permitted by its outputs, its reduction factor will be smaller that the one allowed by a saturated counter with the same number of outputs.

Moreover, if a counter is not-simple, it has at least one input, with weight 2^i greater than one, which will contribute to the counter capacity with its weight. In other words it is equivalent to 2^i inputs of the lower order, 2^0. The reduction factor shall therefore be smaller than the reduction factor of a simple-(saturated)-counter.

Note that in fig. 6 (4,3) (5,3) (6,3) counters are used together with saturated (7,3) counters only when necessary.

The implementation of non-simple counters can be done, as shown in chapter III, using PROM suitably programmed.

For example, the (1,4,3; 4) counter of fig. 10 *b* can be obtained using a $256 \times 4 = 2^8 \times 4 = 1024$ bits PROM.

Parallel counters can be linked to form more complex types. Meo [10] has given a general theory of networks of counters and a synthesis procedure for minimizing the number of counters in a multiplier.

Its results, applied to (3,2) and (7,3) counters, are illustrated in fig. 14.

Fig. 14 *a, b* shows how two or more counters can be connected using Meo's rules: the result thus obtained is to reduce to *one* the number of output bits of weight 2^0 and to produce a set of carries in the following column of 2^1 weight.

These carries are then associated to the initial columns bits, and reduced in the same way, using as many counters (linked as in fig. 14 *a, b*) as necessary.

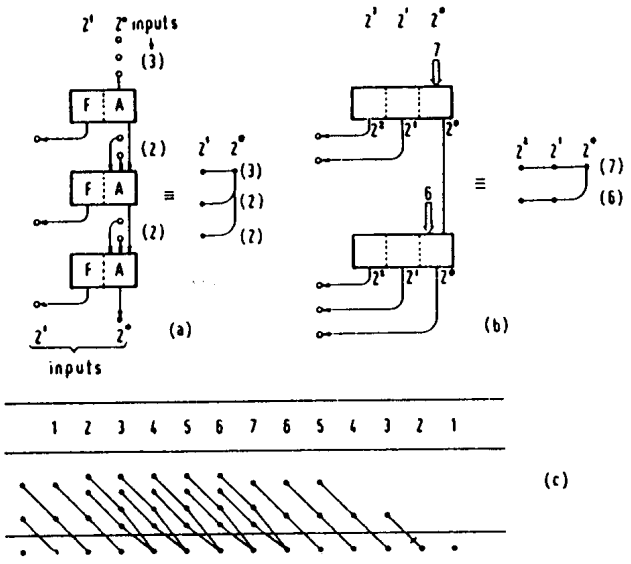

Fig. 14. - *a*) A network of (3,2) counters obtaining a single output bit of weight 2^0 and a set of carries of 2^1 weight; *b*) idem, using (7,3) counters; *c*) using the above composite counters in a 7×7 bits multiplier (compare to fig. 1).

The resulting scheme is a variation of the basic scheme of fig. 1, and although it offers a minimized solution, in terms of counter's number, it affords a large delay, as carriers are propagated through chains of linked counters. Moreover, the saving in counters number appears, over the methods followed in figs. 3, 5, 6, quite modest.

Conclusions.

Some schemes for obtaining fast digital multipliers, based on the reduction of the set of summands equivalent to the product to two numbers whose sum equals the product (this being then obtained by means of a parallel adder), are reviewed with the aim of suggesting an implementation using available LSI integrated circuits (particularly ROMs with small access time). It is shown that a simple counter with 7 inputs and three outputs can be implemented with a ROM, and it is also suggested that two or more counters could be realized in a single chip in order to reduce the count of packages.

Special counters are also considered, useful in reducing the complexity of a multipier in some particular cases.

It can be estimated that, using available components, parallel multiplier for words lengths commonly used (e.g. 16, 32, etc.) can be implemented with a total delay of about 100 ns. or less.

The paper was received on June 16, 1976.

References

[1] I. Flores: *The logic of computer arithmetic.* Prentice Hall, Englewoods Cliffs, N. J., 1963.

[2] C. G. Wallace: *A suggestion for a fast multiplier.* « IEEE Trans. on Electronic Computers », vol. EC-13, Feb. 1964, p. 14 ÷ 17.

[3] L. Dadda: *Some schemes for parallel multipliers.* « Alta Frequenza », vol. XXXIV, n. 5, May 1965, p. 349 ÷ 356.

[4] J. C. Hoffmann, B. Lacaze, P. Czillag: *Multiplieur parallèle à circuits logique itèratifs.* « Electron. Lett. », vol. 4, May 1968, p. 178.

[5] R. De Mori: *Suggestion for an i.c. fast parallel multiplier.* « Electron. Lett. », vol. 5, Feb. 1965, p. 50-51.

[6] H. H. Guild: *Fully iterative fast array for binary multiplication and fast addition.* « Electron. Lett. », vol. 38, May 1969, p. 843-852.

[7] A. Habibi, P. A. Wintz: Fast multipliers. « IEEE Trans. on Comput. », vol. C-19, Feb. 1970, p. 153-157.

[8] W. J. Stenzel: *A class of compact high-speed parallel multiplication schemes.* Dept. Computer-Science, Univ. of Illinois at Urbana, Techn. Rep. UIUCDCS-R-75-756, Sept. 1975.

[9] D. Ferrari: *Un moltiplicatore numerico parallelo sperimentale.* In: Rendiconti della 67ª Riunione Annuale dell'A.E.I., Sept. 196, vol. 3, n. 29, p. 1-4.

[10] A. R. Meo: *Arithmetic networks and their minimization using a new line of elementary units.* « IEEE Trans. on Computers », vol. C-24, n. 3, March 1975, p. 258-280.

[11] D. Ferrari, R. Stefanelli: *Some new schemes for parallel multipliers.* « Alta Frequenza », vol. XXXVIII, n. 11, Nov. 1969, p. 843-852.