MULTIPLE ADDITION OF BINARY SERIAL NUMBERS

Luigi Dadda

Politecnico di Milano
Istituto di Elettrotecnica ed Elettronica
Piazza Leonardo da Vinci, 32 -20133 Milano

Abstract

It is shown how circuits for the addition of several serial binary numbers can be obtained as a combination of parallel counters and memory cells.

The various schemes belong to one of three different classes, characterized by the way in which carries, produced by parallel counters, are treated.

A comparison is made between the various schemes, in terms of speed and complexity.

Introduction

The addition of several binary numbers as a single operation is desirable in many cases in order to increase the speed of computation.
The most important case is given by the multiplication, obtained through the addition of the multiplicand, logically anded by the bits of the multiplier and suitably shifted. A number of schemes based on such principle have been proposed and are well known.
In a recent paper [1], Swartzlander, Gilbert and Reed have shown how a complex computing unit capable of performing the addition of several products would be necessary in some applications. Such a computing unit (the "inner product" computer) is shown to be conveniently implementable by representing one of the factors in serial form, and adding in a single stage all the binary serial numbers composing the various products: with $\underline{m}$ product terms and $\underline{n}$ bits multipliers, there are $\underline{m \times n}$ numbers to be added.
The purpose of this paper is to show how the addition of several binary serial numbers could be performed, and to classify and compare the different possible schemes, in term of speed and complexity.
This problem has been considered in the past, at the writer's knowledge, mainly for numbers represented in parallel.
The case of serial multipliers has been considered in [2] by Dadda and Ferrari and by Swartzlander in [4]. Multiple addition for parallel numbers has been considered by Ho and Chen [5], where they use the same "reduction scheme" proposed in [1]; Singh and Waxmann [3] proposed a scheme which can be considered as belonging to one of the classes of schemes (class 3) that will be illustrated.

+) Work supported by CNR, Consiglio Nazionale delle Ricerche, Rome, and performed at Centro di Studio sulla Ingegneria dei Sistemi per la Elaborazione delle Informazioni, c/o Politecnico of Milan.

Logical schemes for multiple adders

Let us consider $\underline{m}$ binary numbers, each composed by $\underline{n}$ bits, each of the bits of the $\underline{m}$ numbers having the same weight, appearing simultaneously. The schemes to be described are intended to produce their sum, also as a serial number. The sum shall be composed by $\underline{n} + \|\underline{m},\underline{n}\|$ bits, $\|\underline{m},\underline{n}\|$ being the number of bits necessary to represent the integral part of: $\underline{m}(1-2^{-\underline{n}})$. Moreover, the least significant bit (L.S.B.) of the sum could be delayed by δ clock intervals with respect to the L.S.B. of the addends. Ideally, it should be $\delta = 0$, but, due to the complexity of the circuit necessary for obtaining the sum, the corresponding delay could be larger than the clock interval. In such a case, in order also to obtain the sum as a series of pulses synchronized with the addends, it could be convenient to implement the adder with intermediate synchronized memory cells, so that the L.S.B. of the sum is delayed by some (δ) clock intervals.

In the following, a series of logical schemes shall be illustrated, based on different criteria. Use will be made of $(\underline{p};\underline{q})$ "parallel counters" (widely used in parallel multipliers [1]) they are combinational circuits having $\underline{q}$ outputs and $\underline{p} < 2^{\underline{q}}$ inputs, the $\underline{q}$ outputs representing, if interpreted as a binary number, the number of inputs whose value is "one" (see fig 1 for their graphic representation). More general types of counters can be used, as the ones in fig 1,b): they can have inputs with different weights and/or more than one output having the same weight.

In the following, "simple" counters shall be used, see fig 1,a): they are said "saturated" if : $\underline{p}=2^{q}-1$.

Class 1 adders: feed back-carries schemes.

The simplest adding procedure is represented in fig. 2,a): it is the well known way of obtaining the addition "by hand" starting with the leftmost column, counting the "ones" in it and writing the result (as a binary number) on a diagonal, in order to put the carries under the appropriate columns; then the second column is treated in the same way (taking into account the carry from the first column), etc.
This procedure can be "mechanized" by means of a device performing the counting of a column's "ones", and recording the result into memory cells as shown, the L.S.B. representing the bit of the sum, the remaining bits (the carries) being recorded under the next columns. If the counter is stepped to the left at each clock signal, the successive sum's bits are obtained.
A different and equivalent way of representing the same procedure is given in fig. 2,b), where the counter is fed at its inputs by as many shift registers as there are numbers to be added and carries to be accounted for.
Fig 2,c) represents, finally, the circuit, composed by a parallel counter and a suitable number of memory cells, composing short shift registers , fed by the carries outputs from the parallel counter.
Note that in fig.2, $\delta=0$, since the counter's delay is assumed to be smaller than the clock period.Note also that ,after the last, most significant addend's bits have been applied, more columns have to be fed to the circuit, in order to produce the most significant bits of the sum. These columns are obtained by repeating the sign bit (the last, leftmost) of each number, since negative numbers are assumed to be represented as 2-complements.
The basic scheme of fig. 2,a) has been considered for the implementation of parallel multipliers [1], but it has been noted that a large delay could result due to the propagation of carries through the various counters connected in cascade. This objection is not valid in a serial adder, where the carries must be delayed by one or more clock intervals.

The fig 2 scheme assumes that a parallel counter with a sufficient number of inputs and a sufficiently small delay (as compared to the clock interval) can be implemented. Note that, being $\underline{m}$ the number of the addends, the number of carries, $\underline{c}$, is the smallest integer satisfying the condition:

$$2^{\underline{c}+1}-1 \geqslant \underline{m}+\underline{c}$$

and that the counter shall therefore have:

$$\underline{p}=\underline{m}+\underline{c} \quad \text{inputs and:}$$

$$\underline{q}=\underline{c}+1 \quad \text{outputs.}$$

Assume now that parallel counters with a limited number $\underline{p}<(\underline{m}+\underline{c})$ of inputs are to be used. Two different solutions are then possible.
The first is to add the outputs of $\underline{g}$ (>1) ($\underline{p}$,$\underline{q}$) counters (using a parallel adder) so that a composite counter is obtained with a sufficient number of inputs.
A less expensive and faster circuit is obtained by partitioning the $\underline{m}$ addends into $\underline{g}$ subsets in such a way that with a single counter the sum of a subset of addends is obtained; see fig. 3, where (7;3) counters are assumed, and $\underline{g}=3$.
In other words, the circuit reduces the original set of addends to an equivalent set of $\underline{g}$ numbers. By using an adding circuit fed by these $\underline{g}$ numbers one can then obtain the desired sum (if $\underline{g}$ is large enough, more reduction stages would be necessary). In the example of fig. 3, a single (5;3) counter is sufficient for the scope.
Some remarks follow on the fig.3 scheme, where it has been assumed $\underline{m}=13$ and counters with $\underline{q}=3$ are used. It should be noted that the carries from the counters (to be delayed by one or two clock intervals) are applied to the same counter's inputs: they could equally well be applied to a different counter, without affecting the result. A large variety of equivalent circuits can then be obtained: see in fig. 3,d) an example, where one counter is fed only by the carries produced by the other counters.
If the total delay in fig. 3,c)or d) is too large, one could insert memory elements at the interface between the two cascaded stages. This would cause the operation of the circuit to be split into two consecutive clock intervals, thus obtaining a "pipe-line" with a higher clock

frequency.
A case of particular interest is the one using (3;2) counters, i.e. full adders, see fig.4.
In such a case the number $\underline{g}$ of (3;2) counters necessary in the first reduction stage is:

$$\underline{g} = \underline{m}/2 \qquad (\underline{m} \text{ even})$$

$$\underline{g} = (\underline{m}-1)/2 \qquad (\underline{m} \text{ odd}).$$

The set of $\underline{g}$ ($\underline{m}$ even) or ($\underline{g}$+1) ($\underline{m}$ odd) numbers thus obtained can be reduced by a similar circuit, until a single output, i.e. the desired sum, is obtained. It can be shown that $\lceil \log_2 \underline{m} \rceil -1$ reduction stages are required ($\lceil \underline{x} \rceil$ representing the smallest integer larger than $\underline{x}$).
If memory elements are placed at the interfaces of adjacent stages, a pipeline structure is obtained with a frequency of operation determined by the sum of the delay of a full adder and of a memory element, and $\delta = \lceil \log_2 \underline{m} \rceil - 1$.
Note, also, that the structure is a tree of full adders, each with a carry memory and another memory at the sum output. If a delay larger than the one afforded by a single full adder and memory element is tolerated, inter-stage memories can be inserted every two or more stages, thus reducing δ and decreasing the frequency of operation; or, in order to obtain an equivalent result, counters with a greater number of inputs can be used, requiring a smaller number of memory cells.
Table $\underline{A}$ contains some parameters of adders designed for $\underline{m}$ = 256, using counters of various size. It can be shown that if the counters are implemented by a network of full adders (see [6]), the total numbers of full adders used in the different solutions are only slightly different. It can be seen from the Table that the total number of carry memories decreases as the counter's inputs increases.
Moreover, the total delay (expressed as a multiple of a full adder delay) increases with the counter's inputs. Note also that the delay of the least significant output bit is smaller than the delay of the remaining output bits.(In counters implemented by ROMs this is not true. On the other hand, such counters are suitable only for a small number of inputs, and if a number of them are combined in order to obtain a counter of a larger size, the overall output's delays are again different).
As in all schemes of class 1 and class 2 (but not in those of class 3) the output bits of weights 2,4,8,etc. have to be delayed by 1,2,3, etc. clock intervals, respectively, it should be possible to design counters in such a way that such delays include those of the combinational part of the counters, so that the effective delay to be accounted for, is the one of the L.S.B. only (i.e. the minimum).

Further investigation is necessary to determine the best way of designing parallel counters for this application: for sake of brevity this will not be done here.

Class 2 adders: feed-forward carries schemes.

In the preceding paragraph, all schemes have in common the characteristic of being composed by parallel counters whose carries outputs are fed back to the counter's inputs via suitable delays.
A second class of schemes can be obtained by not feeding back the carries to the counter's inputs, but considering the counter's outputs as a set of numbers equivalent to the set of addends (this is the principle on which the reduction schemes used in parallel multipliers are based, see [1]).
This set can be further reduced until a single number, equivalent to the addend's sum, is obtained.
The procedure is illustrated by the fig.5,a) example, where $\underline{m}$ = 13. In the first stage, a (13;4) counter is used, obtaining four numbers, whose sum is the desired sum. The second stage reduces this set of four numbers to an equivalent set of three numbers, using a (4;3) counter; the third stage obtains an equivalent set of two numbers, by means of a (3;2) counter, and finally a single number, i.e. the desired sum, is obtained by means of the last stage, which is the only one requiring a feedback carry.
Fig 5,b) is the scheme of the various counters used, where the first one is fed in parallel by the $\underline{m}$ addends, and fig. 5,c) represents the corresponding logical scheme; fig.5,d) is an equivalent scheme .
It can be seen immediately that this class of adders compares unfavorably with the preceding class 1 adders.
First of all, while in the class 1 schemes the operation can in principle be obtained by means of a single counter, this is not possible, with the present scheme, as it appears from fig. 5. The combinational part of the counters is nevertheless not very different, since, for a given number $\underline{m}$ of addends, the single counter necessary in the fig. 2 scheme requires $\underline{m}$ + $\underline{c}$ inputs (c is the number of carries), while in fig. 5 requires $\underline{m}$ inputs only. On the other hand, in fig. 5 some more cascaded counters are needed, whose complexity decreases very rapidly through the various stages.
Moreover, the number of memory cells necessary in fig. 5 schemes is slightly larger than those necessary in fig. 2 schemes.
The main drawback of class 2 schemes in comparison to class 1 schemes is the larger number of stages necessary, which gives a larger delay.
As has been done for class 1 schemes, also for class

2 schemes, counters with a limited number of inputs can be used. For instance, using (3;2) counters, each stage has a number of outputs which is $\sim 2/3$ the number of inputs: note that in class 1 schemes (fig. 4) the outputs of a stage are $\sim 1/2$ the number of the inputs.
For the preceding reasons, the class 2 schemes shall not be considered any further.

Class 3 adders, with carries summed in parallel adders.

In class 1 and class 2 schemes, the carries produced by the counters are delayed (see fig. 2 and 5) until they appear in the appropriate columns. Instead of being simply shifted, such carries can undergo a suitable processing. Adders, which uses such possibility, are grouped in a new class, class 3.
Among the various possibilities of handling carries, the following two appear of particular interest for their simplicity.
The first class 3 scheme is given in fig. 6, where the $\underline{q}$ counter's outputs are associated with the two ($\underline{q}$-1) bits numbers, stored in two registers, and representing the result of the operation of the circuit in the preceding step.By means of ($\underline{q}$-1) full adders the three numbers are reduced to a two-number set. The sum-output of the rightmost full adder represents the bit of the sum, while the remaining output bits of the full adders (and the leftmost bit of the main ($\underline{p}$;$\underline{q}$) counter) represent the new carries, to be stored as shown in the registers by means of the clock pulse.
With reference to fig.5, a) the above procedure corresponds to adding, with a carry-save adder , the successive outputs of the main counter, representing the contribution to the sum from the various columns.
A second scheme is given in fig. 7, which is based on the same principle of the fig. 6 scheme, with the difference that the addition of the successive counter's outputs is performed through an ordinary parallel adder (possibly with carry look-ahead). The "lenght" λ of the adder is related to $\underline{m}$, the number of addends , by:

$$\lambda = \left[\log_2 \underline{m}\right]$$

while in fig. 6 scheme one stage less is required.
Fig. 8, a) and b), represents, by an equivalent logical circuit, the same schemes in fig. 6 and 7, respectively. Similar circuits have been already proposed in [2] for multipliers with a parallel multiplicand and a serial multiplier.
As far as the complexity is concerned, classs 3 adders are generally better than class 1 (and, of course, class 2) adders.
If counters are implemented as a network of full adders, class 3 schemes require (for the same $\underline{m}$) a slightly smaller number of full adders than class 1 schemes.
The number of memory cells in class 3 schemes is considerably smaller than for class 1 schemes , even if for the latter a solution requiring the smallest number of memory cells is considered.
For the Table $\underline{A}$ example ($\underline{m}$ = 256) 34 is such number for a class I scheme, while for the same $\underline{m}$ a class 3 scheme of the type given in fig. 7 requires only 9 memory cells, and 16 are needed for a fig.6 scheme.
As far as the effect of the delays in the counters is concerned, it has been noted that in counters implemented with full-adders they increase from the L.S.B. to the M.S.B. outputs.
In both fig. 6 and fig. 7 schemes, the bit sum appears with a further full-adder delay with respect to the L.S.B. of the counter. The clock necessary to store the new carries must be applied after the M.S.B. has appeared: note that this clock could be suitably delayed with respect to the clock used to store the sum bit, in order to increase the overall operating frequency of the adder.

Concluding remarks

The problem of adding several binary serial numbers has been consiaered;, with the purpose of identifying and comparing different schemes. Three classes of schemes have been proposed, all being based on parallel counters used to sum simultaneous bits (column summing).
In the first class of schemes the carries generated by a single counter are delayed and associated with the following columns for counting. The single counter can be partitioned in a set of smaller counters, and if (3;2) counters or full-adders are considered, a very simple and straightforward scheme is obtained.
In the second class of schemes, the set of counter's outputs(being equivalent to the original set of addends and composed of a smaller number of elements) is first obtained. This set can be further reduced by more counters, until the final sum is reached. This class of schemes is shown not offering any advantage over the other two.
In the third class, the counter outputs are immediately summed, in a carry-save adder or in a carry-look-ahead adder, with the carries from the preceding column.
It is interesting to compare the above schemes for multiple serial addends with those used for parallel addends [1].In the last case, minimum delay is achieved by using reduction schemes (used for class 2 adders) not requiring any carry propagation, which takes place only at a final stage consisting of a carry-look-ahead adder. Carry propa-

gation does not constitute a major problem in adders, as has been shown, so that those schemes which must be avoided in parallel adders can be adopted, leading to an acceptable solution, as in class 1 adders. Reduction schemes, i.e. class 2 adders, on the other hand , offer a larger delay and a greater complexity in serial adders.

References

1 - L.Dadda: "Some schemes for parallel multipliers" Alta Frequenza, vol.34, n.5, pp.349-356, May 1965.

2 - L.Dadda, D.Ferrari: "Digital multipliers: a unified approach", Alta Frequenza, vol.37, n.11, pp.1079-1086, nov. 1968.

3 - S.Singh, R.Waxmann: "Multiple operand addition and multiplication", IEEE Trans.on Comp., vol.C-22, n.2, pp.113-120 , febr. 1973.

4 - E.E. Swartzlander: "The quasi serial multiplier" IEEE Trans.on Comp., vol.C-22,n.4, pp.317-321, april 1973.

5 - J.T.Ho,T.C.Chen: "Multiple addition by residue threshold functions and their reprentation by array logic", IEEE Trans. on Comp., vol. C-22, n.8, pp.762-767, aug. 1973.

6 - E.E. Swartzlander: "Parallel counters", IEEE Trans. on Comp., vol. C-22,n.11, pp.1021-1024, nov. 1973.

7 - E.E. Swartzlander, B.K.Gilbert, I.S. Reed: "Inner product computers", IEEE Trans.on Comp., vol. C-27, n.1, jan. 1978.

Table A: Feed-back-carries adders for m = 256, using counters of different size

counter's size	n.° of counters	n.° of carries feedback memories	n.° of stages	counters composed by full adders	
				max. number of full adder delays	number of full adder delays for the L.S.B.
(3;2)	255	255	8	8	8
(7;3)	64	192	4	12	8
(15;4) (5;3)	23 1	141	3	13	8
(31;5)	10	100	2	14	8
(63;6) (32;6)	4 1	75	2	18	8
(127;7) (16;5)	2 1	52	2	17	9
(255;8) (9;4)	1 1	34	2	18	9
(262;9)	1	36	1	14	6

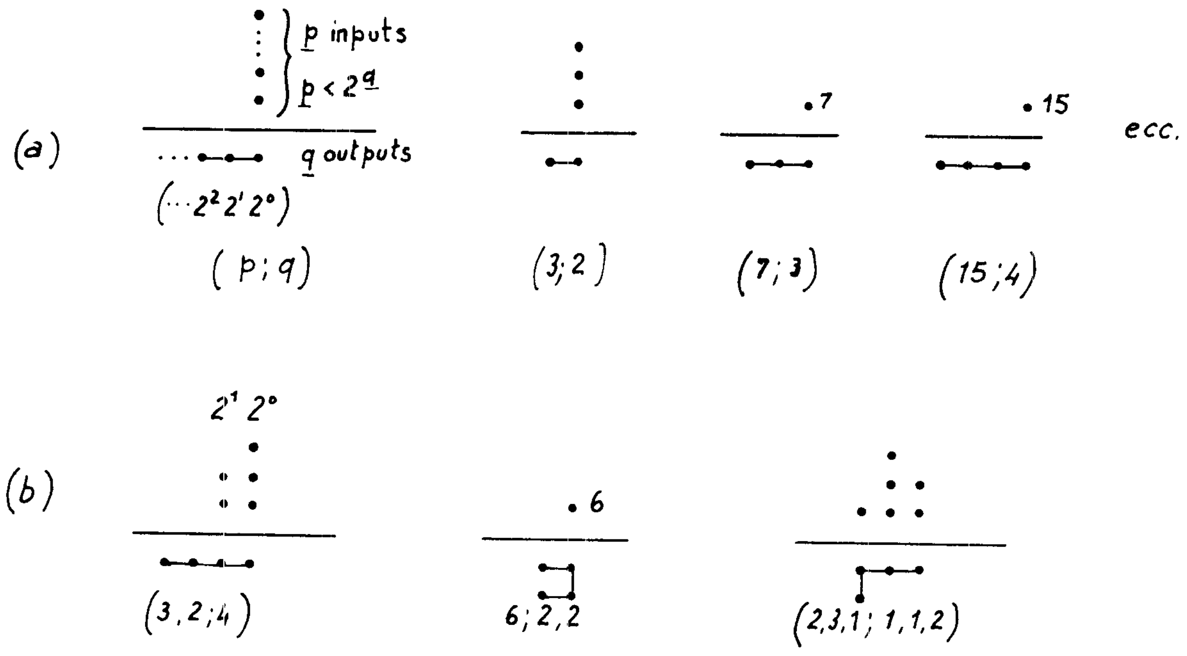


Fig. 1 : Graphical representation of parallel counters.

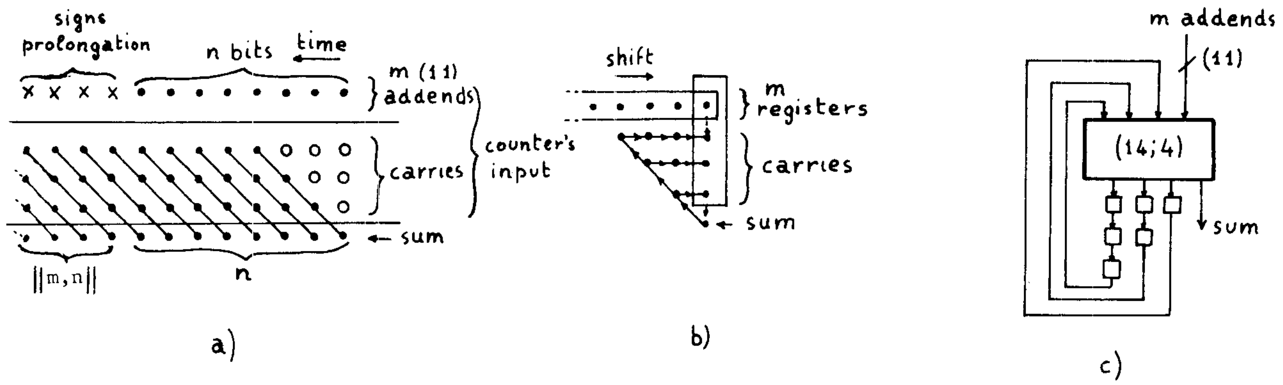


Fig. 2 : Basic schemes for class 1 adders (carries feed-back schemes) assuming parallel counters with unlimited number of inputs.

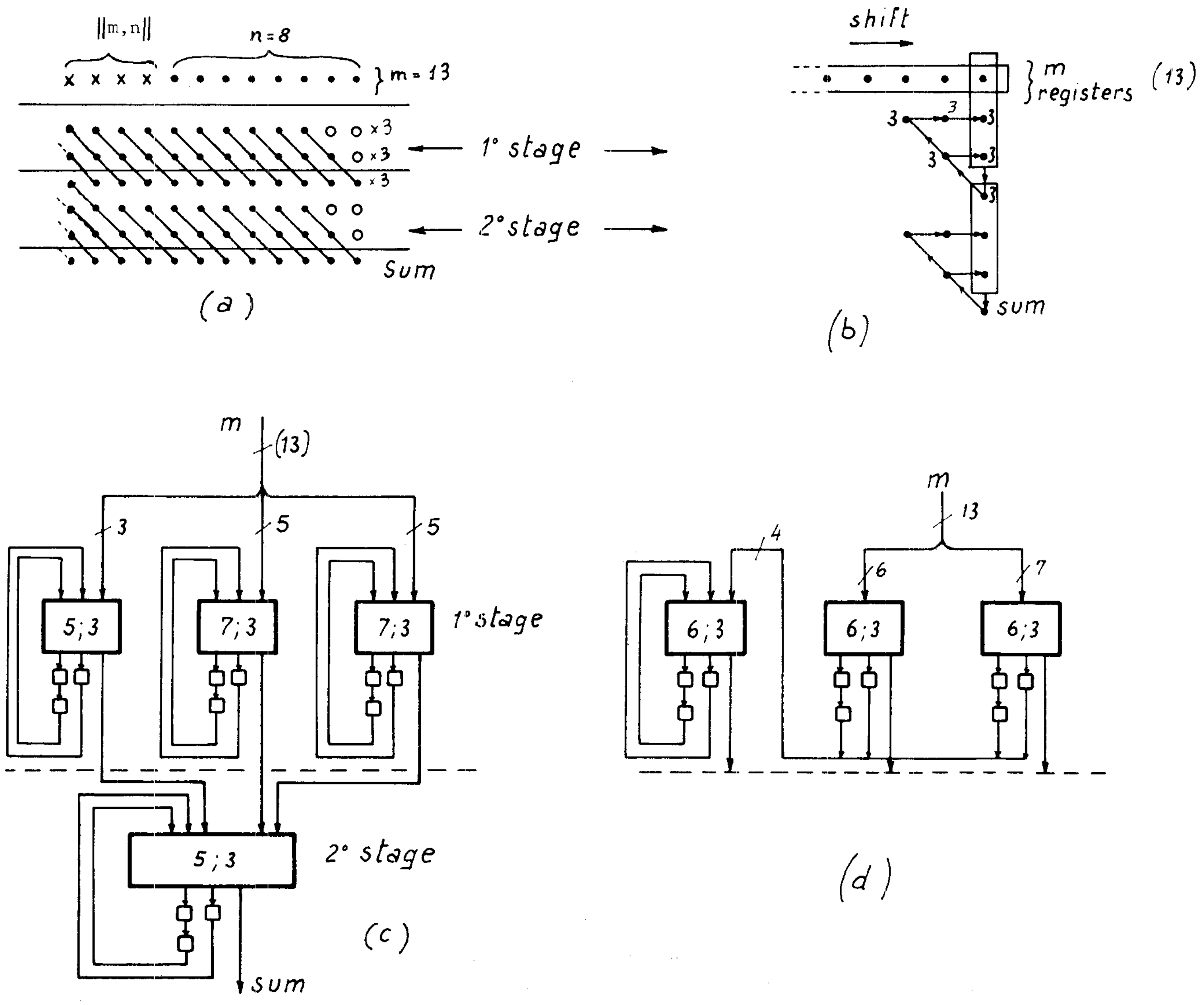


Fig. 3 : Schemes of class 1 adders, implemented by means of parallel counters with a prescribed number of inputs.

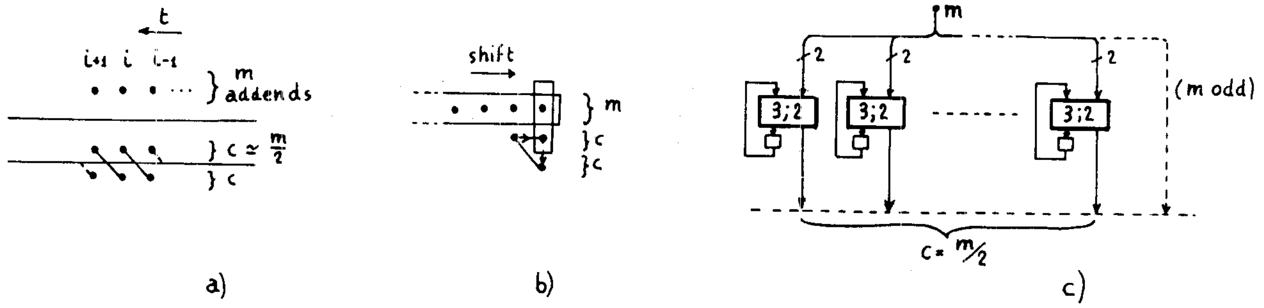


Fig. 4 : Schemes of class 1 adders, using only (3;2) counters (i.e. full adders).

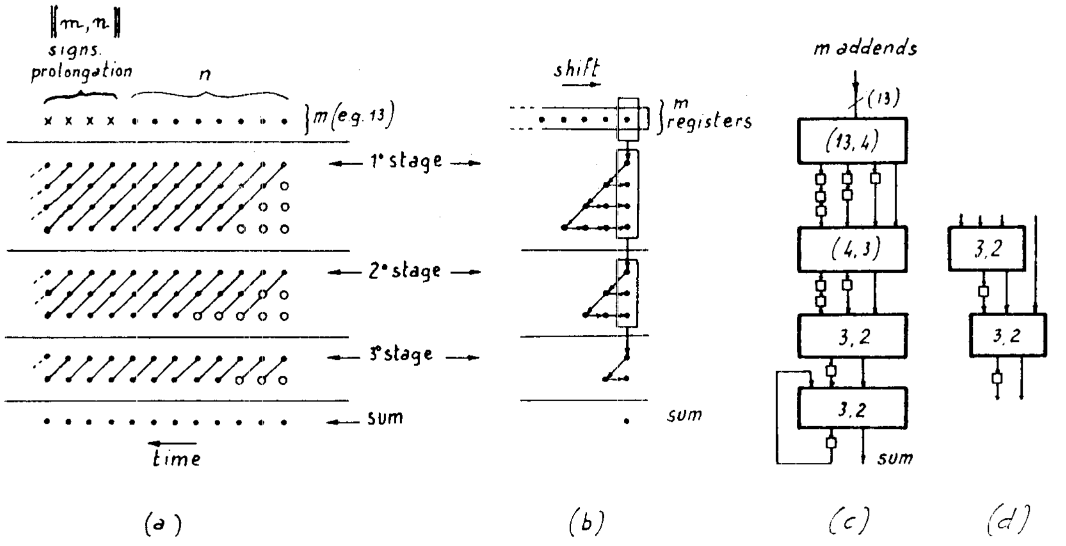


Fig. 5 : Basic schemes for class 2 adders (carries feed-forward schemes) assuming parallel counters with unlimited number of inputs.

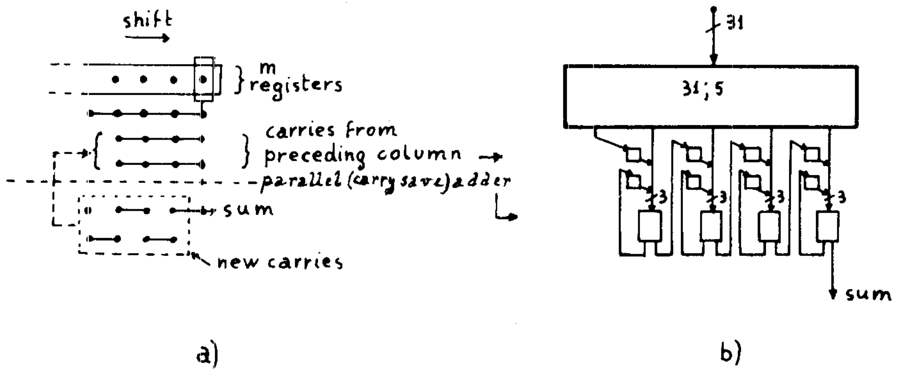
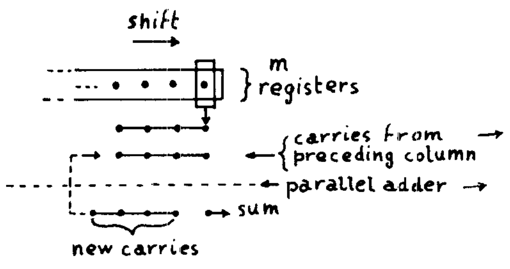
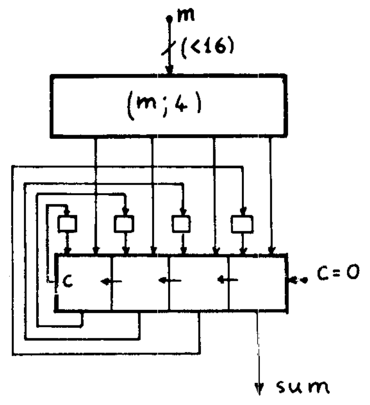


Fig. 6 : Schemes for a class 3 adder, where carries are summed in a carry-save parallel adder.

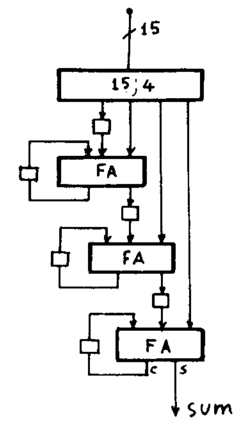


a)

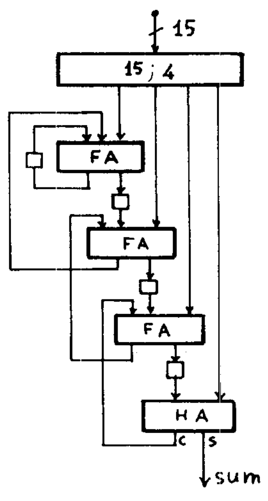


b)

Fig. 7 : Schemes for a class 3 adder, where carries are summed in an ordinary parallel adder.



a)



b)

Fig. 8 : A different representation of fig.6 (a) and fig. 7 (b) schemes.