THREE CELL STRUCTURES FOR TERNARY CELLULAR ARRAYS

J.A. Bate and J.C. Muzio

Department of Computer Science
University of Manitoba
Winnipeg, Manitoba

Three cell structures are given for use in ternary cellular arrays. Universal arrays for both general functions of n variables and symmetric functions are considered, and the optimum size is found. Arrays are presented for these cases, as well as for sum and carry functions.

1. Introduction

There has been much interest in recent years in the realization of combinational switching functions using cellular arrays. An overview of the topic up to 1970 may be found in reference 4. Akers[1] introduced a cellular array for binary functions. It consisted of a rectangular array of identical cells, each having two inputs and outputs as well as an external control input which determines the action of the cell (see fig. 1 a). The edge inputs were fixed, and the two outputs of each cell were the same, both being set to the value of one of the inputs, depending on the control input as shown in fig. 1 b.

One possible generalization of these arrays to ternary logic was given by Miller and Muzio[3]. The ternary cells were cubical cells each with three inputs and outputs. Like the Akers arrays, the edge inputs were fixed, and the three outputs of each cell were identical.

A somewhat different binary array was proposed by Kukreja and Chen[2], In this array, the edge inputs were not fixed, but were available for use in realizing functions, and the two outputs of each cell were not the same. The action of these cells is shown in fig. 1 c.

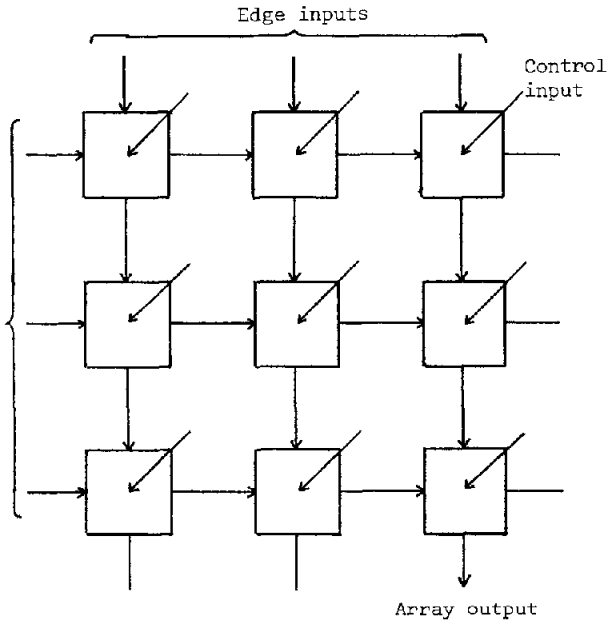


Fig. 1 a) A binary cellular array

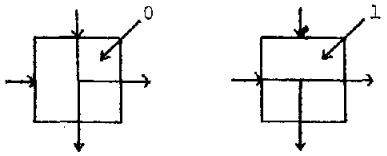


Fig. 1 b) Akers' cells

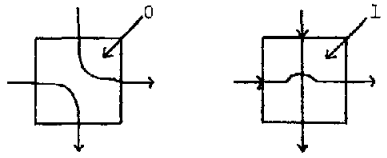


Fig. 1 c) Kukreja and Chen's cells

In this paper, three generalizations of these binary cells will be considered. The three cell types will be described, and the formation and evaluation of arrays will be discussed. Universal arrays, symmetric universal arrays and arrays realizing the sum and carry functions for ternary arithmetic will then be given. In the former two cases, the optimum size is found, and arrays which achieve the optimum are given for most cases.

2. The Cells

The cells to be considered are all cubical cells with three inputs (top, rear, and left) and three outputs (bottom, front, and right). Each cell is controlled by a single ternary input, which determines the action of the cell. The three types differ in the switching actions they perform. The actions are shown in fig. 2., and given in table form in fig. 3.

The inputs and outputs are numbered 0, 1, and 2, referring to the front/rear, left/right, and top/bottom directions, respectively. The values of the inputs are denoted i_0 i_1 and i_2 .

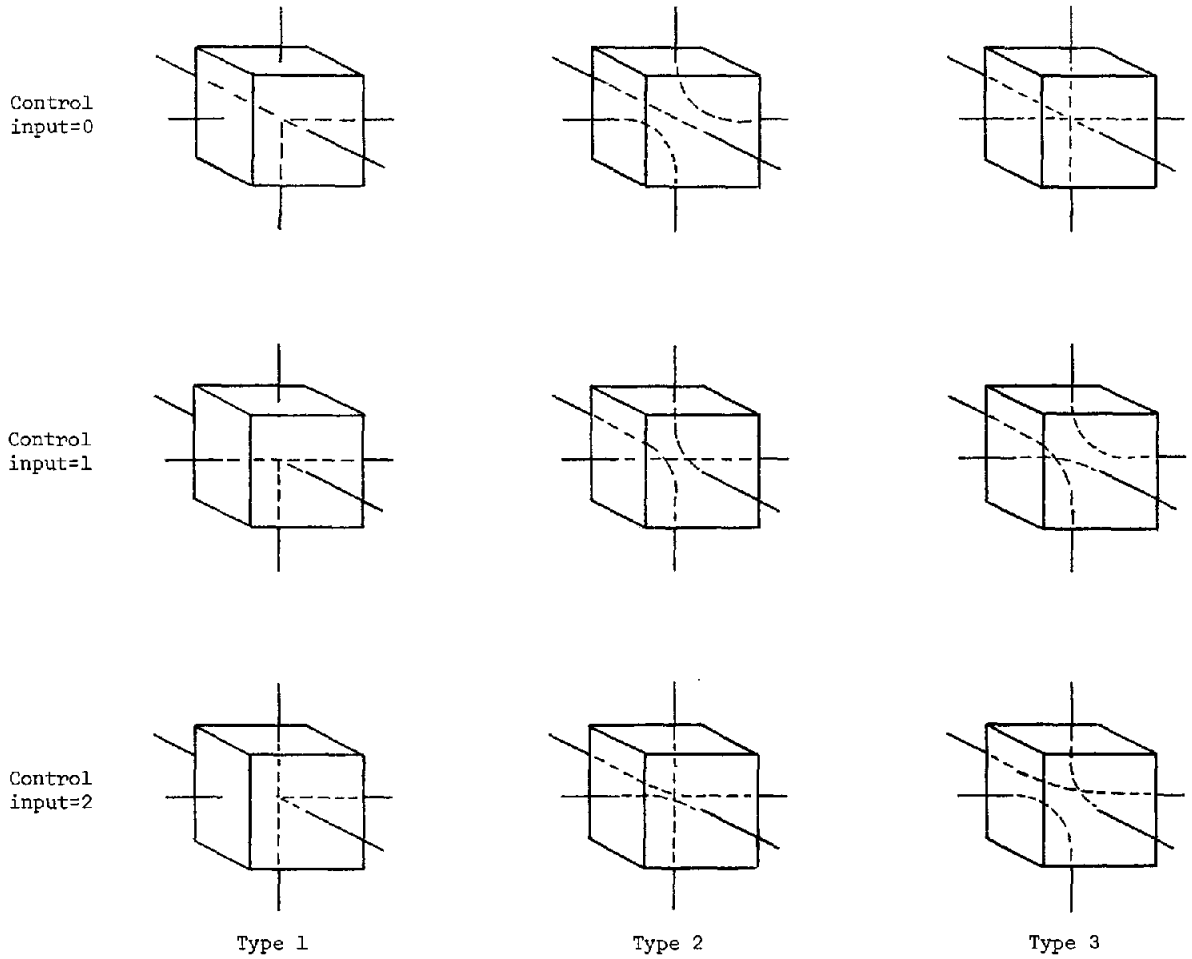


Fig. 2 The three cell types

Control input	Outputs Type 1 0 1 2	Type 2 0 1 2	Type 3 0 1 2
0	$i_0\ i_0\ i_0$	$i_0\ i_2\ i_1$	$i_0\ i_1\ i_2$
1	$i_1\ i_1\ i_1$	$i_2\ i_1\ i_0$	$i_1\ i_2\ i_0$
2	$i_2\ i_2\ i_2$	$i_1\ i_0\ i_2$	$i_2\ i_0\ i_1$

Fig. 3 The switching actions

The first type of cell is another possible generalization of the Akers cell. The three outputs are the same, all being set to one of the inputs, depending on the control input, as shown.

The second and third types are possible generalizations of the Kukreja and Chen switching cells. In these types, the three outputs are different. If the cells are restricted to symmetrical cells in which each input is routed to each of the outputs for some control value, these two are the only possible types.

For type 3 cells, one of the control input values (0) causes the inputs to pass straight through and so the other two values must cause the inputs to cycle clockwise and counterclockwise. For type 2 cells each control value causes one input to pass straight through, and the other two to switch.

3. Evaluation of Arrays

The output of an array of cells will be the front output of the lower right front cell. Each control and edge input may be a constant, a variable, or a Post (cyclic) negation of a variable (fig. 4).

A	$\bar{A}$	$\bar{\bar{A}}$
0	1	2
1	2	0
2	0	1

Fig. 4

Post Negations

In this way, an array may be used to realize any combinational ternary function. For example, the sample array in fig. 5., using type 1 cells, finds the sum digit of A, B, and C.

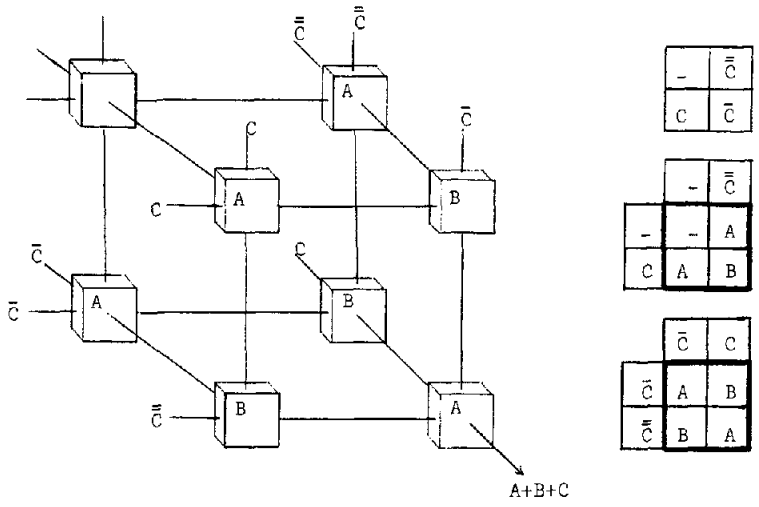


Fig. 5 A sample array

Arrays will be represented as shown on the right of fig. 5. The edge and control inputs are given plane by plane from top to bottom, with the boundary of the array itself given by a bold face line so that control and edge inputs will not be confused.

There are two ways to evaluate the output of an array for a specific set of values. The first is to begin at the output and trace the path back through the array to find the edge input from which the output will come. The cells through which the path travels are known as "pivot" cells.

For example, to evaluate the array in fig. 5 for A = 0, B = 1, and C = 2, begin at the output. The control input of the first cell is A = 0 so the output will come from the rear input of that cell. The next pivot cell is then the cell behind it. The control input there is B = 1, and so the next pivot cell will be to the left. Finally, that cell has control A = 0, and so the edge input at the rear of that cell will determine the value of the array output. That edge input is $\bar{\bar{C}} = 2 = 0$.

For type 2 and 3 cells, it is also necessary to consider which output of each pivot cell was used, as well as the control value, to determine the next pivot cell.

Alternately, the evaluation procedure could consist of the repeated deletion of layers of the array. Beginning with the output cell, as before, the input of that cell which will become the output can be found. If it is, say, the left input, it is clear that the right hand layer of the array may never be reached again, and may be deleted. The new pivot cell is now the lower right front cell again, and the new array may be evaluated in the same manner (remembering that the output is now from the right side of that cell, instead of the front).

In algorithmic form:

1) The first time, the output is the front output.
2) Evaluate the control value of the lower right front cell, and determine which input becomes the output.
3) Delete the appropriate layer of the array (bottom, right, or front for the top, left, or rear input, respectively). Remember the new direction.
4) Repeat steps 2 and 3 until an edge input is reached.

This approach is well suited to some computer implementation of the evaluation procedure.

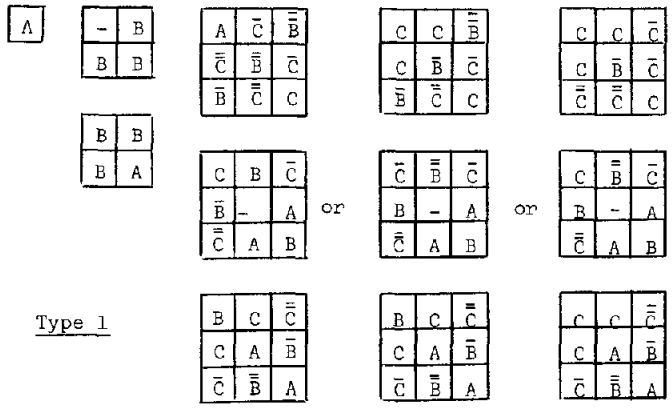
4. Universal Arrays

A universal array for n variables is a single array which can be used to realize any ternary function of n variables. It is sufficient to find an array, controlled by the n variables, which will select a different edge input for each possible set of values which the variables may take. Any desired function may then be obtained by setting these edge inputs to the appropriate function values.

First consider the optimal size for such an array. There must be at least 3^n edge inputs, and so a cubical array k cells on a side must have $3k^2 \geq 3^n$ or $k \geq 3^{\frac{n-1}{2}}$. For an optimal array, then, $k = \lceil 3^{\frac{n-1}{2}} \rceil$. For odd n , this is exact, and we may have a cubical array with exactly the correct number of edge inputs. For even n , there is no simple pattern, but $k = 2 \times 3^{\frac{n-2}{2}}$ is an adequate approximation to the optimum value. It is exact for $n \leq 4$, and this value for k will be considered "optimum" for all practical purposes.

Small optimal universal arrays may be found

fairly easily for $n = 1$ and 2, and without too much trouble for $n = 3$. For $n = 4$, a $6 \times 6 \times 6$ array is required which is not easily found except for type 2 cells as noted below. Optimal arrays for small n are given in fig. 6. The edge inputs are the 3^n function values. The positions of the inputs for each set of values are not given, but they may easily be found.



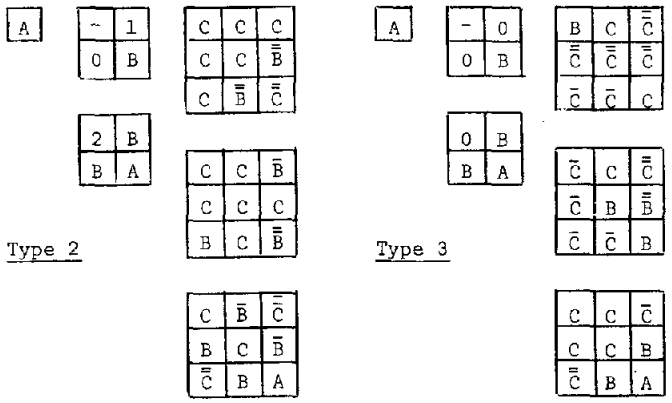


Fig. 6 Optimum universal arrays for small n

For type 2 cells, we may extend the given arrays inductively to obtain optimal arrays for any n. This is because these cells have the property that a cubical block of identical cells will perform the same function on planes of inputs and outputs as a single cell does on single inputs and outputs. For example, a block of 0's will pass all the back inputs straight through, while the left inputs will be routed to the bottom and the top inputs to the right.

This property, plus the fact that the only cell containing the variable A is the lower right front cell for the arrays for type 2 cells given in fig. 6, enables arrays to be found for any n. Using the 3-variable array as the "A" cell, arrays for 4 and 5 variables may be obtained by using $3 \times 3 \times 3$ blocks of cells with the same patterns as for 2 and 3 variables. The 5 variable array may be used as the basis for 6 and 7 variable arrays, etc.

No such apparent induction is evident for types 1 or 3.

For all three types there exist simple non-optimal arrays for all n. These are cubical arrays 2^{n-1} cells on a side, and the arrays for small n are given in fig. 7. For small n, these arrays may actually be preferable to the optimal arrays due to their symmetry and regularity, which are, after all, the properties most desired in cellular arrays.

It is interesting to note that the non-optimal arrays for type 2 cells are much more complex than the ones for the other types, whereas type 2 cells have a simple optimal form.

5. Symmetric Functions

A symmetric function is one for which the output will be the same for any permutation of the inputs. A symmetric universal array is an array which can realize any symmetric function of n variables. The optimal size for such an array is clearly smaller than for a general universal array, since only $\binom{n+3}{3}$ edge inputs are required rather than 3^n. For example, the three sets of values (0 0 1) (0 1 0) and (1 0 0) give the same function value, and so they may (and should) use the same edge input. The number of required edge inputs is not the critical factor in determining the optimal size for these arrays. The optimal size of a symmetric universal array for n variables is $n \times n \times n$ as shown below.

Theorem: The smallest regular size for a symmetric universal array for n variables is $n \times n \times n$.

Proof: Every path from the output to an edge input which is selected by some set of values for A, B, C, ... must pass through at least one cell controlled by each of A, B, C, ...

Trace a path beginning at the output. It must eventually pass through a cell controlled by A, at which point one value for A will direct the path to the left. Similarily, a cell must be reached controlled by each of B, C, ..., and in each case some value will direct the path one layer left.

Therefore, for at least one set of values, the edge input selected must be at least n layers to the left of the output. Similarly, there must be at least one edge input n layers above the output, and behind the output.

Therefore, the minimal size is $n \times n \times n$.

This optimum is obtained for all n for cell types 1 and 2. The arrays for $n = 3$ are given in fig. 8, and the pattern is clear. Type 3 cells do not easily form symmetric arrays. Apart from the universal arrays for $n \leq 3$, which are also optimum for symmetric functions, none are known.

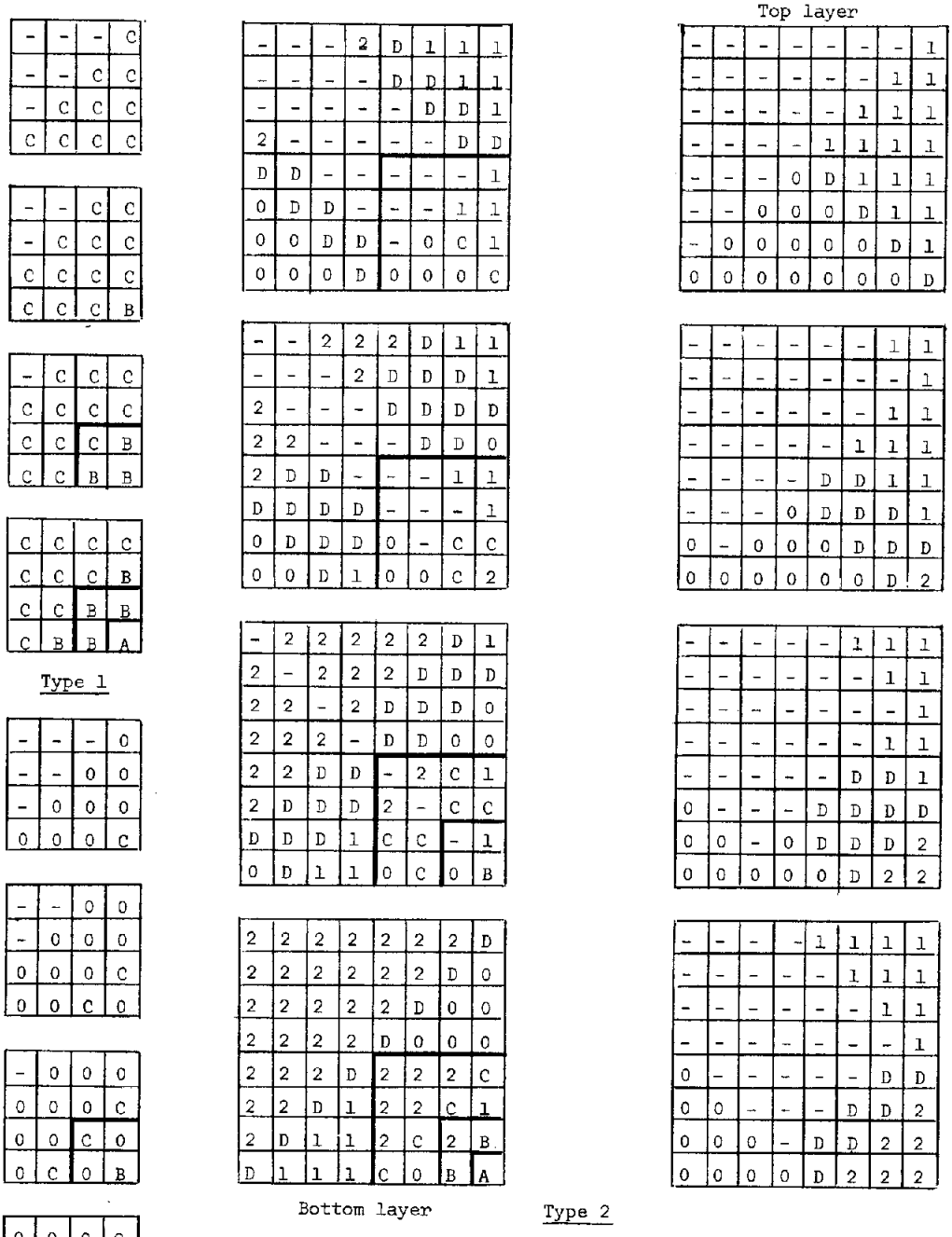


Fig. 7 Other universal arrays. A 4-variable array is shown for type 2 because of its complex pattern. Arrays for smaller numbers of variables are sub-arrays of the arrays given, and are marked with bold lines.

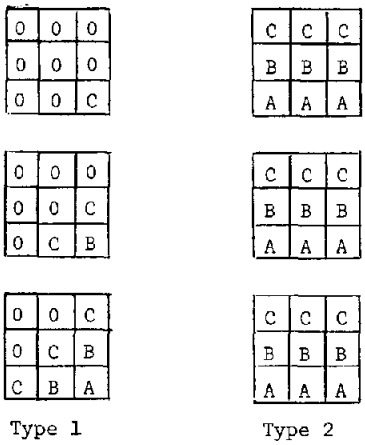


Fig. 8 Symmetric universal arrays

6. Sum and Carry Arrays

Efficient arrays may also be constructed to realize specific functions, such as the sum and carry functions for three ternary variables. The sum functions may be realized by using the two-variable universal arrays, since once two values are known, the sum will be a Post negation of the remaining variable, which is an allowable edge input.

The carry functions do not reduce as nicely, but for type 2 and 3 cells may still be realized with $2 \times 2 \times 2$ arrays. Type 1 cells require a $3 \times 2 \times 2$ array due to the fact that the three outputs of each cell are identical.

The sum and carry arrays for each of the three types are given in fig. 9.

7. Summary

The three cell types presented are all ternary generalizations of the binary cells of Akers, Kukreja, and Chen. In each case, arrays may be constructed to realize any given ternary function.

Type 2 cells seem to be the most useful in almost all respects. Optimal universal arrays exist for all n, as well as optimal symmetric universal arrays. Compact arrays to implement specific functions are easily found.

Type 1 cells have the advantage of being simpler in operation. Optimal symmetric universal arrays exist for all n, but optimal general universal arrays are known only for $n \leq 3$, and arrays for specific functions will usually be larger than for the other two types. Type 3 cells have no apparent advantage over the other two. Optimal arrays for both general and symmetric functions are known only for $n \leq 3$.

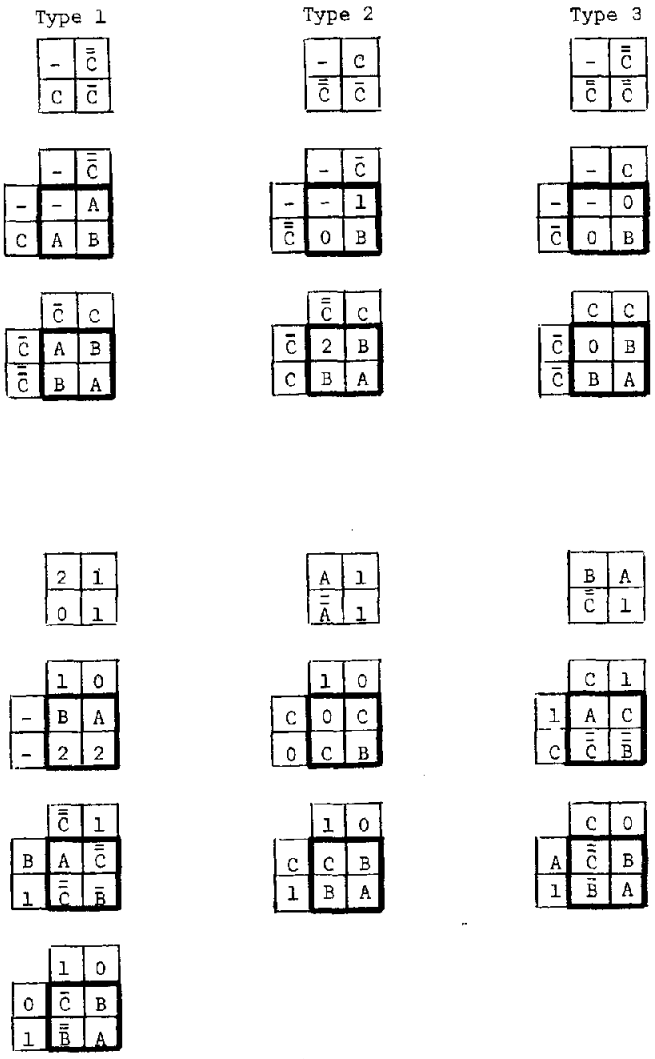


Fig. 9 Sum and carry arrays

References

[1] S.B. Akers, "A rectangular logic array", IEEE Trans. Comput., Vol. C-21, pp. 848-857, 1972.

[2] S.N. Kukreja and I-Ngo Chen, "Combinational and sequential cellular structures", IEEE Trans. Comput., Vol. C-22, pp. 813-823, 1973.

[3] D.M. Miller and J.C. Muzio, "A ternary cellular array", Proc. 1974 International Symp. on Multiple-Valued Logic, pp. 469-482.

[4] A. Mukhopadhyay and H.S. Stone, "Cellular logic", in "Recent Developments in Switching Theory" (A. Mukhopadhyay, ed.), pp. 255-313, Academic Press, 1971.