TWO-DIMENSIONAL ITERATIVE LOGIC*

Rudd H. Canaday
Bell Telephone Laboratories, Incorporated
Whippany, New Jersey

INTRODUCTION

It is well known that given a suitable Boolean function, a large number of "gates" or "elements," each producing this function, can be interconnected in a regular structure, or "array," to realize any given Boolean function. Furthermore, the structure of the array can be invariant to the function being realized.

One of the simplest such structures is the two-dimensional array of three-input one-output elements shown in Fig. 1. In this paper two methods are presented for using this structure in the synthesis of arbitrary Boolean functions. The following assumptions will be adhered to throughout this paper:

1. All elements in the array are identical.
2. The interconnections between elements in the array are fixed. They cannot be broken or changed in any way.
3. The array will be used as a single output circuit. Only the output of the lower right element of the array is accessible to the outside world.
4. Every element in the array realizes the "majority" function

$$f(A,B,C) = AB + AC + BC$$

of its three inputs.*

As a consequence of assumptions (1), (2), and (4), an array can be described completely in terms of its width w and height h. Such an array will be called a "MAJority Array" or "MAJA."

In the remainder of this paper it will be shown, first, how to synthesize an arbitrary "self-dual" function in a MAJA. Then this result will be extended to arbitrary functions and some examples will be given. This is "intersection synthesis." Next a second synthesis technique, "factorization synthesis," will be described, first in a canonic form, through examples, and then in a more general form.

*The material presented in this report is based on a thesis submitted in partial fulfillment of the requirements for the Doctor of Philosophy Degree in Electrical Engineering at the Massachusetts Institute of Technology, September 1964.
The research reported was made possible through the support extended to the M.I.T. Electronic Systems Laboratory by the U.S. Air Force Avionics Laboratory, Navigation and Guidance Division, under Contract AF-33(657)-11311 and, in the earlier phases of this research, under Contract AF-33(657)-8932.

*It is easy to prove[1] that all of the results given here extend directly to arrays of "minority" elements:

$$f(A,B,C) = \overline{AB + AC + BC}$$

This paper is based on the author's Ph.D. thesis.[1] In the present paper space limitations preclude statements of all theorems and proofs on which the synthesis methods are based. These do appear, together with extensions of the results presented here, in reference 1.

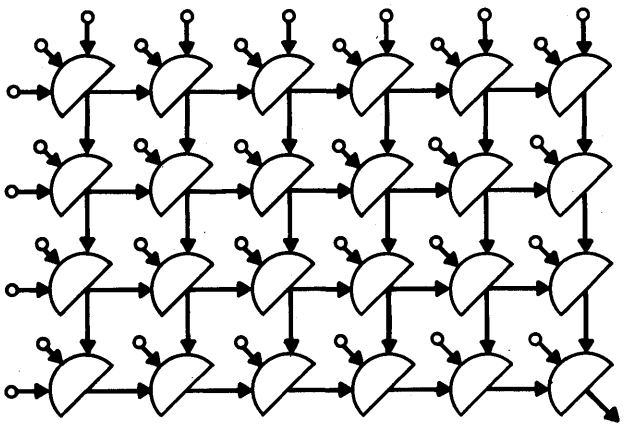

Figure 1. A 4 × 6 array of 3-input elements.

Both synthesis techniques lead to arrays of reasonable size, and embody new synthesis techniques which may prove to be applicable in other forms of synthesis also.

PRELIMINARY DISCUSSION

Before discussing array synthesis, it is necessary to define some terminology for arrays.

Each element in an array has three inputs, which will be denoted "top," "center," and "left" inputs (signal flow in an array is always left-to-right and top-to-bottom).

The w inputs (for an array of width w) consisting of the top input to each element in the top row of the array form the "top boundary" inputs to the array.

Similarly, the h inputs (for an array of height h), consisting of the left input to each element in the leftmost column of the array, form the "left boundary" inputs to the array.

One particular type of array proves to be of particular interest. This array has, in effect, all top boundary inputs wired together, and all left boundary inputs similarly wired together.

Definition : An " XY Standard Boundary Condition majority array" (XY SBC MAJA) is a MAJA all of whose top boundary inputs carry the signal Y where Y can be a variable or a constant, and all of whose left boundary inputs carry the signal X, where X can be a variable or a constant.

Fig. 2 is an example of an XY SBC MAJA. The two synthesis methods to be presented both apply to the SBC MAJA.

Intersection synthesis is given first for self-dual functions, as defined below.

Definition: Given a Boolean function $f(x_1, \ldots, x_n)$, then the *dual* $f^d(x_1, \ldots, x_n)$ of the function f is defined as:

$$f^d(x_1, \ldots, x_n) = f(x_1, x_2, \ldots, x_n)$$

By applying deMorgan's theorem one can easily see that if f is expressed using only the operations $+$ (OR), $\cdot$ (AND) and $-$ (NOT), then f^d is obtained by interchanging $+$ and $\cdot$ throughout the expression for f.

Definition: A Boolean function $f(x_1, \ldots, x_n)$ is *self-dual* if and only if

$$f^d(x_1, \ldots, x_n) = f(x_1, \ldots, x_n)$$

Note that by this definition of dual and self-dual, a function which is a constant is not self-dual since, if $f \equiv 1$ then $f^d = f \equiv 0$.

Any n-variable Boolean function $f(x_1, \ldots, x_n)$ can be factored as

$$f(x_1, \ldots, x_n) = Xf_0 + Yf_1 \qquad (1)$$

with X and Y chosen from $\{x_1, x_1, \ldots, x_n, x_n\}$.

If f is a self-dual function, then the existence of the factorization (1) implies that f can be factored as

$$f = Xf_0 + Yf_0{}^d + XY \qquad (2)$$

where X, Y, and f_0 are the same as in Eq. (1).

Equation (2) is basic to the synthesis algorithm, which is presented in the following two definitions and Theorem 1 below.

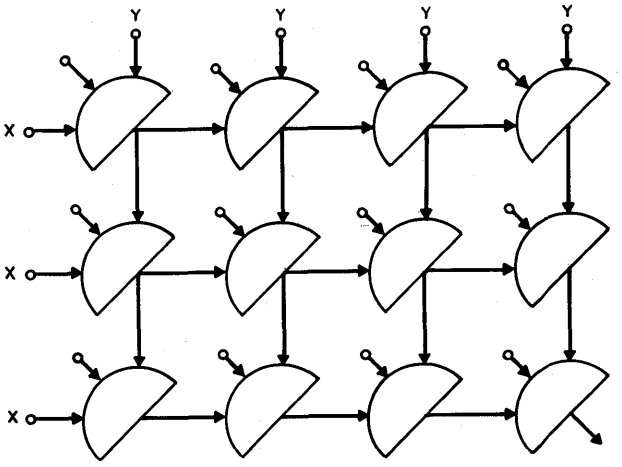

Figure 2. An XY SBC MAJA.

INTERSECTION SYNTHESIS

Definition: Given two Boolean functions f_a and f_b, and given a sum of products expression for each: $f_a = r_1 + r_2 + \ldots + r_k$; $f_b = t_1 + t_2 + \ldots + t_m$, then an intersection matrix of $f_a \times f_b$ is a matrix with k rows and m columns, in which each entry e_{ij} is the intersection of the literals in r_i with the literals in t_j (i.e., e_{ij} contains a literal y if and only if y is in both r_i and t_j).

Note that the intersection matrix for a given f_a and f_b is not unique. It is unique for given sum-of-products expressions (including the ordering of their terms) for both f_a and f_b. Now it is possible to define an SBC MAJA to realize any given self-dual function.

Definition: Given a self-dual function $f^{sd} = XY + Xf_0 + Xf_0{}^d$, and given a $k \times m$ intersection matrix $f_0 \times f_0{}^d$, with rows corresponding to terms of f_0, then an *XY intersection MAJA* for f^{sd} is a $k \times m$ XY SBC MAJA with the center input to the ij^{th} element chosen to be any one of the literals in entry e_{ij} of the intersection matrix, for all i, j: $1 \leq i \leq k$, $1 \leq j \leq m$.

Again note that one function f^{sd} may have many intersection MAJAs for each factorization (each choice of X and Y).

Theorem 1: Given any XY intersection MAJA for a self-dual function f^{sd}, then the output of the MAJA realizes the function f^{sd}.

*Proof:** By construction the MAJA has no constant inputs. Therefore it is sufficient to prove that the MAJA produces all the ones of f^{sd}, since a MAJA without constant inputs must realize a self-dual function.[1] It is easy to prove that if the term XY is one the array output is one. Now let a term Xr_1 in Xf_0 be one. Then every literal in the term is one. Then every left boundary input, and the center input to every element in the i^{th} row, is one. It is not difficult to prove that this condition suffices to insure that the array output is one. Thus the array output is one for every term in Xf_0. Similarly if a term Yt_i in $Yf_0{}^d$ is one then every center input to the i^{th} column, as well as every top boundary input, is one. Again this suffices to insure that the array output is one. Thus every one of $f^{sd} = XY + Xf_0 + Yf_0{}^d$ is realized at the output of an intersection MAJA for f^{sd} and so the MAJA realizes f^{sd}.

The synthesis algorithm just presented allows one to synthesize any self-dual Boolean function. To extend the result to any arbitrary Boolean function, the "self-dual expression" for a function is defined.

Definition: Given any n-variable Boolean function $f(x_1, \ldots, x_n)$, and a variable U independent of $(x_1, \ldots, x_n)$, the *Self-Dual Expression* f^{sd} for f is defined as the $(n+1)$-variable function:†

$$f^{sd}(U,x_1,\ldots,x_n) = Uf(x_1,\ldots,x_n) + \bar{U}f^d(x_1,\ldots,x_n).$$

*The proof given here is very sketchy. The detailed proof, which depends on a number of theorems not given here, is in reference 1.

†This is a reformulation of work done by S. B. Akers.[2]

It is trivial to prove that the self-dual expression for any function is a self-dual function. Also, if f is a selfdual function, then

$$f^{sd}(U,x_1, \ldots, x_n) = f(x_1, \ldots, x_n).$$

Clearly this is true if and only if f is self-dual.

To synthesize an arbitrary n-variable function, proceed as follows:

1. Find the $(n + 1)$-variable self-dual expression, f^{sd}, for the function f.
2. Synthesize $f^{sd}(U,x_1, \ldots, x_n)$.
3. Replace every input U to the array by the constant input 1 (one) and every input $\overline{U}$ by the constant 0 (zero).

The resulting array realizes $f(x_1, \ldots, x_n)$ since $f^{sd}(1, x_1 \ldots, x_n) = f(x_1, \ldots, x_n)$ by construction.

The examples to follow show arrays with the inputs U and $\overline{U}$. Thus these arrays, as shown, realize the self-dual expression of the given function. Wherever the inputs U and $\overline{U}$ occur, they can be replaced by 1 and 0 as discussed above to obtain the array for the given function.

Note that the array for a self-dual function contains, by construction, no constant inputs. It can be shown that in any MAJA constant inputs are required if and only if the function being synthesized is non-self-dual.

In an intersection MAJA for a function, every term in the factored expression for the function corresponds to a single row or column in the MAJA. It can be shown[1] that terms in the output function of an SBC MAJA can correspond not only to single rows and columns, but also to inputs (or elements) which do not form a single row or column. Thus it seems that the intersection matrix construction does not make maximum use of the MAJA. In other words, by realizing some terms in the function using a set of elements not from a single row or column, it is possible to realize many functions in an SBC MAJA considerably smaller than an intersection MAJA for the function. By extensions to this work, reduced non-SBC arrays can be derived also, but the methods become much messier and less algorithmic.

It is not possible in the space available here to discuss reduction techniques. However, the following examples show some arrays in reduced form, as well as the original intersection arrays.

While it is possible to construct an intersection array from any factorization of the form

$$f^{sd} = XY + Xf_0 + Yf_0^d$$

with f_0 and f_0^d each expressed as a sum of product terms, it is obvious that the smallest array results from choosing X and Y and the expressions for f_0 and f_0^d to minimize the number of terms in f_0 and in f_0^d. This is done in the following examples.

SYNTHESIS EXAMPLES

Before giving examples of synthesis by Theorem 1, it is useful to define a notation which will be used in examples throughout the rest of this work.

In the many examples to follow in this and succeeding sections it is necessary to show arrays with variables assigned to the inputs. Since the interelement connections in an array are fixed, an array with inputs can be completely specified by giving each boundary input and the center input to each element of the array. These inputs are presented as a matrix, with a line separating top and left boundary variables from the center input variables. Thus the array of Fig. 3 is represented by

	B	B	B	B	B
$\overline{B}$	$\overline{U}$	C	C	$\overline{D}$	A
$\overline{B}$	D	$\underline{C}$	$\underline{C}$	U	$\underline{A}$
$\overline{B}$	C	$\overline{A}$	$\overline{A}$	$\overline{D}$	$\overline{C}$
$\overline{B}$	$\underline{C}$	$\overline{A}$	$\underline{A}$	$\overline{U}$	$\overline{C}$
$\overline{B}$	$\overline{C}$	$\overline{D}$	$\overline{U}$	$\overline{D}$	$\overline{C}$

Clearly this representation is completely general; it is not restricted to SBC arrays.

Example 1: $f(A,B,C,D)=\Sigma\emptyset,1,4,6,7,8,11,12,13,14.$*

A minimum Sum of Products (MSP) form of the Self-Dual Expression for this function is:

$$f^{sd}=BCDU + ABCDU + ABCD + ABCD + ABC + ABCU + ABCU + ABCDU + BDU + CDU$$

This function can be factored on $\overline{B}B$ or $\overline{C}C$ without increasing the number of product terms (1Ø) in the expression, since the term $B\overline{D}U$ can be written $BC\overline{D}U$ or $\overline{C}\overline{D}U$ can be written $\overline{B}\overline{C}\overline{D}U$, without changing f^{sd}. Arbitrarily choose the $\overline{B}B$ factorization.

*This notation, defined in Caldwell,[3] defined the rows of the truth table for which the function is one.

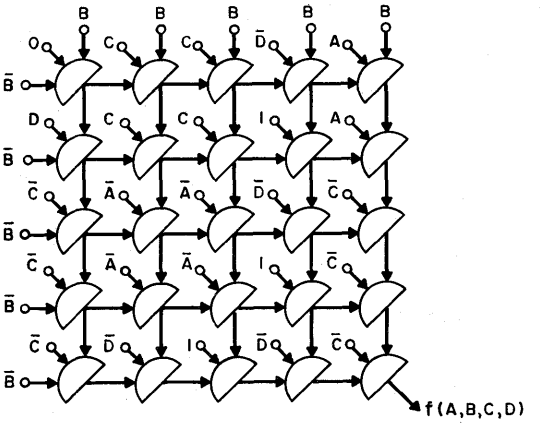

Figure 3. SBC MAJA for Example 1.

The intersection matrix is

	B $(\bar{C}D\bar{U})$	$(\bar{A}C\bar{D})$	$(\bar{A}CU)$	$(\bar{D}U)$	$(A\bar{C})$
$(AC\bar{D}\bar{U})$	$(\bar{U})$	$(C\bar{D})$	(C)	$(\bar{D})$	(A)
$(ACDU)$	(D)	(C)	(CU)	(U)	(A)
$(\bar{A}\bar{C}\bar{D})$	$(\bar{C})$	$(\bar{A}\bar{D})$	$(\bar{A})$	$(\bar{D})$	$(\bar{C})$
$(\bar{A}\bar{C}U)$	$(\bar{C})$	$(\bar{A})$	$(\bar{A}U)$	(U)	$(\bar{C})$
$(\bar{C}\bar{D}U)$	$(\bar{C})$	$(\bar{D})$	(U)	$(\bar{D}U)$	$(\bar{C})$

One intersection MAJA is

	B	B	B	B	B
$\bar{B}$	$\bar{U}$	C	C	$\bar{D}$	A
$\bar{B}$	D	C	C	U	A
$\bar{B}$	$\bar{C}$	$\bar{A}$	$\bar{A}$	$\bar{D}$	$\bar{C}$
$\bar{B}$	$\bar{C}$	$\bar{A}$	$\bar{A}$	U	$\bar{C}$
B	$\bar{C}$	$\bar{D}$	U	$\bar{D}$	$\bar{C}$

By reduction techniques† described elsewhere,[1] a 2×3 non-SBC array can be found to realize this function:

	C	$\bar{B}$	$\bar{D}$
D	$\bar{U}$	$\bar{A}$	U
A	A	$\bar{C}$	B

A 2×3 array is known to be the smallest array capable of realizing this function since no smaller array has enough terminals. The 2×3 array shown here has one element which performs no logical function because it has two identical (A) inputs. The 5-element network resulting from removal of element 21 has the adsolute minimum number of 3-input elements for any network capable of realizing this function.

†These techniquees are heuristic, and results obtained depend to some extent on the experience of the person doing the reduction.

Example 2: $f(A,B,C,D) = \Sigma 1,4,5,6,7,9,11,12,13,15$

MSP $f^{sd} = BD + \bar{A}\bar{C}D + AB\bar{C} + \bar{A}BU + ADU$ can be written as

$$f^{sd} = BD + B(A\bar{C}+\bar{A}U) + D(\bar{A}\bar{C}+AU)$$

The SBC MAJA is

	D	D
B	A	$\bar{C}$
B	U	$\bar{A}$

This is the smallest SBC MAJA which can possibly realize this function, since six different literals must appear as inputs, and no smaller SBC array has six inputs.

Example 3: $f(A,B,C,D) = \Sigma 1,2,5,7,11$

$$\text{MSP} \quad f^{sd} = \bar{A}\bar{C}D + \bar{A}BD + A\bar{B}CD + \bar{A}B\bar{C}\bar{D} + AB\bar{C}D\bar{U} + \bar{B}D\bar{U} + CD\bar{U} + \bar{A}B\bar{U} + \bar{A}C\bar{U}$$

Factor on $\bar{D}D$ for the minimum number of product terms in the factored expression

$$f^{sd} = \bar{D}(\bar{A}\bar{B}C+AB\bar{C}\bar{U}+\bar{A}\bar{B}\bar{U}+\bar{A}C\bar{U}) + D(\bar{A}\bar{C}+\bar{A}B+A\bar{B}C+\bar{B}\bar{U}+C\bar{U})$$

A corresponding SBC intersection MAJA is:

	D	D	D	D	D
$\bar{D}$	$\bar{A}$	$\bar{A}$	C	$\bar{B}$	$\bar{C}$
$\bar{D}$	$\bar{C}$	B	A	$\bar{U}$	$\bar{U}$
$\bar{D}$	$\bar{A}$	$\bar{A}$	$\bar{B}$	$\bar{U}$	$\bar{U}$
$\bar{D}$	A	A	C	$\bar{U}$	$\bar{U}$

By reshuffling rows and colums, it becomes possible to remove the row corresponding to term $\overline{A}C\overline{D}\overline{U}$ and the column corresponding to term $CD\overline{U}$

	$\overline{D}$	$\overline{D}$	$\overline{D}$
D	$\overline{A}$	$\overline{C}$	$\overline{A}$
D	A	B	A
D	C	A	$\overline{B}$
D	$\overline{B}$	$\overline{U}$	$\overline{U}$

The term $CD\overline{U}$ is realized by the center inputs to elements 31, 42, and 43, and the left boundary. The term $\overline{A}C\overline{D}\overline{U}$ is realized by the center inputs to elements 11, 21, 31, 42, and the top boundary. This is the smallest known SBC array for this function. However, there exists a 2×3 non-SBC majority array for the function:

	C	C	D
$\overline{B}$	A	$\overline{D}$	$\overline{C}$
$\overline{U}$	$\overline{B}$	$\overline{B}$	$\overline{A}$

Arrays of size 2×3 or 3×3 appear to be typical for non-self-dual functions of four variables.[1]

THE CANONIC ARRAY

The two major disadvantages of the synthesis method presented above are:

1. The lack of reasonable bounds on the size of array needed to realize an arbitrary function.
2. The inability to apply reduction procedures to functions of more than five or six variables.*

The development of a canonic form for arrays for arbitrary functions of n variables as is done below obviates these disadvantages. This canonic form has the following properties:

1. The canonic array for n variables, for a given n, is an array of fixed size, with some inputs fixed and the rest of the inputs chosen for the specific function (typically, well over half of the inputs are fixed). This array will realize any given function of n variables if the nonfixed inputs are properly chosen. An algorithm for determining the inputs needed to realize any given function exists.
2. An algorithm exists for generating the canonic array for any given n.
3. The canonic array for n variables is the smallest known array to realize the checkerboard (worst-case) function of n variables, for n even.
4. For most given functions the canonic array is reducible (by methods given in reference 1).
5. The canonic array embodies a technique for embedding arrays within larger arrays, which shows great promise for future work in multiple output arrays and in nonmajority (nonminority) arrays.

The disadvantage of the canonic array is that the array required for a given function usually is larger than the array produced by intersection synthesis and reduction, assuming that the function is small enough to make that synthesis-reduction technique feasible.†

The size of the canonic array is shown in Table 1 as a function of n , the number of variables in the function to be synthesized. In addition a "connection count" is shown for each n. This is the number of connections to the array which are not invariant over all functions of n variables, plus one connection for each variable whose input connections are invariant. One can envision building the array with all invariant connections wired together at the time of manufacture. Then the "connection count" is just the number of input terminals to the array to allow it to realize any function of n variables.

Table 1.

n	Size	Connections
3	3×3	10
4	4×6	16
5	7×8	26
6	9×14	44
7	15×18	78
8	19×30	144
9	31×38	274

*Note, however, that the basic synthesis algorithm (Theorem 1) can be applied to arbitrarily large functions, though the resulting arrays generally are unreasonably large.

†Reduction is feasible on most functions of five variables and some functions of six variables.[1]

CONSTRUCTION OF CANONIC ARRAYS

In this section the canonic array for n variables is presented through examples. In the next section the process of embedding subarrays in an array is considered more generally.

Consider the MAJA

	U	U
$\overline{U}$	$\overline{A}$	g_1
$\overline{U}$	g_0	A

Array 1

in which g_0 and g_1 are input literals chosen from the set $\{B,\overline{B},U,\overline{U}\}$. This array, as straightforward analysis will show, realizes the self-dual function

$$f^{sd} = U[\overline{A}g_0+Ag_1] + \overline{U}[\overline{A}g_1+Ag_0]$$

which can be any self-dual function of the three variables (A,B,U). If $U = 1$, Eq. (3) becomes

$$f = \overline{A}g_0 + Ag_1 \qquad (4)$$

which, if g_0 and g_1 are chosen from $\{\overline{B},B,0,1\}$, can be any function of the two variables (A,B). Thus Array 1 is the *canonic factorization* array for $n = 2$ variables. It is called a "factorization" array because Eq. (4) is a factorization of the function. It is called "canonic" because it is in a standard form, as will become clear later.

Now consider the MAJA

	U	U	U
$\overline{U}$	$\overline{A}$	g_{11}	$\overline{B}$
$\overline{U}$	g_{00}	B	g_{10}
$\overline{U}$	$\overline{B}$	g_{01}	A

Array 2

If $U = 1$, this array realizes the function

$$f = \overline{A}\overline{B}g_{00} + \overline{A}Bg_{01} + A\overline{B}g_{10} + ABg_{11} \qquad (5)$$

If g_{00}, g_{01}, g_{10}, and g_{11} are chosen from $\{C,\overline{C},0,1\}$, then Eq. (5) can be any function of the three variables (A,B,C). Array 2 is called the canonic factorization array for $n = 3$ variables even though its form differs slightly from the canonic construction to be defined.

It would be possible to continue thus to define arrays to realize any function of n variables for $n = 4,5,6$, and so on. However, if one wishes to define a canonic factorization array for an arbitrary number of variables, it is necessary to define a construction method which will result in the canonic array for any given n. Here the approach taken is inductive. Given the canonic factorization array for $(n-1)$ variables, it will be shown how to construct the array for n variables. This will be done by embedding two arrays for $(n-1$ variables in the factorization array for n variables. Consider first the case of $n = 4$. The array for $(n-1) = 3$ variables is known (Array 2). Take two of them:

	U	U	U
$\overline{U}$	$\overline{B}$	g_{011}	$\overline{C}$
$\overline{U}$	g_{000}	C	g_{010}
$\overline{U}$	$\overline{C}$	g_{001}	B

Array 3

Array 3, with $U = 1$, realizes

$$g_0 = \overline{B}\overline{C}g_{000} + \overline{B}Cg_{001} + B\overline{C}g_{010} + BCg_{011} \qquad (6)$$

	U	U	U
$\overline{U}$	$\overline{B}$	g_{111}	$\overline{C}$
$\overline{U}$	g_{100}	C	g_{110}
$\overline{U}$	$\overline{C}$	g_{101}	B

Array 4

Array 4, with $U = 1$, realizes

$$g_1 = B\overline{C}g_{100} + B\overline{C}g_{100} + B\overline{C}g_{110} + BCg_{111} \qquad (7)$$

Combine Array 3 and Array 4 in the canonic factorization array for $n = 4$

	U	U	U	U	U	U
$\overline{U}$	$\overline{A}$	$\overline{A}$	$\overline{A}$	$\overline{B}$	g_{111}	B
$\overline{U}$	$\overline{B}$	g_{011}	$\overline{C}$	g_{100}	C	g_{110}
$\overline{U}$	g_{000}	C	g_{010}	$\overline{C}$	g_{101}	B
$\overline{U}$	$\overline{C}$	g_{001}	B	A	A	A

Array 5

where dotted lines have been shown only to clarify the construction of the array. It should be emphasized that Array 5 is a 4×6 $\overline{U}U$ SBC MAJA, with no modification of its structure. The array contains subarrays only in the sense that the input pattern to portions of the array can be identified with the input patterns to Array 3 and Array 4.

Array 5, with $U = 1$, realizes the function

$$f = \overline{A}g_0 + Ag_1 \qquad (8)$$

where g_0 and g_1 are the arbitrary 3-variable functions realized by Array 3 and Array 4. To see this, let $U = 1$ (and $U = 0$, of course). Then if $A = 1$, the subarray corresponding to Array 4 (elements 14,15,16,24,25,26,34,35, and 36) has one on its top boundary (elements 14,15,16), and zero on its left boundary (elements 14,24,34). It is not difficult to see that with $A = 1$, Array 5 has the same output as Array 4. Similarly, if $A = 0$, then the subarray corresponding to Array 3 has zero on its left boundary (elements 21,31,41) and one on its top boundary (elements 21,22,23) and it can be shown that Array 5 has the same output as Array 3. Thus Eq. (8) is vertified.

By substitution of Eqs. (6) and (7) into Eq. (8), one obtains

$$f = \overline{A}\,\overline{B}\,\overline{C}g_{000} + \overline{A}\,\overline{B}Cg_{001} + \ldots + A\overline{B}\overline{C}g_{110} + ABCg_{111} \qquad (9)$$

If g_{000} through g_{111} are chosen from $\{\overline{D},D,0,1\}$, then equation (9) can be any function of the four variables (A,B,C,D).

By interchanging rows and columns in Array 5 and then interchanging U and $\overline{U}$ and changing the subscripts on the g inputs appropriately, one obtains the MAJA,

	U	U	U	U
$\overline{U}$	$\overline{A}$	$\overline{B}$	g_{111}	$\overline{C}$
$\overline{U}$	$\overline{A}$	g_{100}	C	g_{110}
$\overline{U}$	$\overline{A}$	$\overline{C}$	g_{101}	B
$\overline{U}$	$\overline{B}$	g_{011}	$\overline{C}$	A
$\overline{U}$	g_{000}	C	g_{010}	A
$\overline{U}$	$\overline{C}$	g_{001}	B	A

Array 6

which realizes the same function, Eq. (9), as does Array 5. This is the *flipped canonic factorization array* for four variables, "flipped" because it corresponds to Array 5 flipped about its main diagonal (and with U, $\overline{U}$ interchanged and the g's renumbered).

To construct the canonic factorization array for five variables one embeds two 4-variable subarrays in a factorization array in exactly the same manner in which Array 5 was constructed. If one uses as subarrays two copies of Array 5, the resulting 5-variable array is 5×12, with 60 elements. If, however, one uses the flipped array, Array 6, the resulting 5-variable array is 7×8 with 56 elements:

	U	U	U	U	U	U	U	U
$\overline{U}$	$\overline{A}$	$\overline{A}$	$\overline{A}$	$\overline{A}$	$\overline{B}$	$\overline{C}$	g_{1111}	$\overline{D}$
$\overline{U}$	$\overline{B}$	$\overline{C}$	g_{0111}	$\overline{D}$	$\overline{B}$	g_{1101}	D	g_{1110}
$\overline{U}$	$\overline{B}$	g_{0100}	D	g_{0110}	$\overline{B}$	$\overline{D}$	g_{1101}	C
$\overline{U}$	$\overline{B}$	$\overline{D}$	g_{0101}	C	$\overline{C}$	g_{1011}	$\overline{D}$	B
$\overline{U}$	$\overline{C}$	g_{0011}	$\overline{D}$	B	g_{1000}	D	g_{1010}	B
$\overline{U}$	g_{0000}	D	g_{0010}	B	$\overline{D}$	g_{1001}	C	B
$\overline{U}$	$\overline{D}$	g_{0001}	C	B	A	A	A	A

Array 7

Again, the dotted lines are included to clarify the construction. Array 7 realizes the function

$$f = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}g_{0000} + \overline{A}\,\overline{B}\,\overline{C}Dg_{0001}\,7 \ \ldots + ABCDg_{1111}$$

It is interesting to note that the canonic factorization array is the smallest array known which realizes the "checkerboard" function $f(A,B,C,D) = A + B + C + D$.* The canonic factorization array for this function is

	U	U	U	U	U	U
$\overline{U}$	$\overline{A}$	$\overline{A}$	$\overline{A}$	$\overline{B}$	$\overline{D}$	$\overline{C}$
$\overline{U}$	$\overline{B}$	D	$\overline{C}$	$\overline{D}$	C	D
$\overline{U}$	D	C	$\overline{D}$	$\overline{C}$	D	B
$\overline{U}$	$\overline{C}$	$\overline{D}$	B	A	A	A

However, for five variables no function is known which cannot be realized in a reduced intersection MAJA smaller than Array 7. It is true in general that no function is known to be "worst case" for n odd, although the "checkerboard" function is always "worst case" for n even.

The construction of canonic factorization arrays for higher values of n is carried out by successive embedding of $(n-1)$-variable flipped arrays, as was just illustrated for $n = 5$. Let H_n denote the height of the canonic factorization array for n variables, and let its width be W_n. Then, by the construction of the canonic factorization array

$$H_3 = 3, \qquad W_3 = 3$$

and

$$H_n = W_{n-1} + 1, \qquad W_n = 2H_{n-1} \qquad \text{for } n > 3$$

These array sizes are tabulated in Table 1. Note that in the canonic factorization array for n variables $\{x_1,x_2,\ldots,x_n\}$, 2^{n-1} of the inputs, the g inputs,

*This function is termed "checkerboard" because its Karnaugh Map representation resembles a checkerboard.

depend on the function being realized, while all other inputs are fixed, independent of the particular function being realized and equal to one of the $2n$ literals $\{x_1,x_1,x_2,x_2,\ldots,x_{(n-1)},U,U\}$ (U, U) being in fact constants, of course), so that if all identical fixed inputs are wired together at the time of manufacture, only $2n + 2^{n-1}$ external connections to the array need be provided. This "connection count" is also tabulated in Table 1.

As a final illustration, the canonic factorization array for $n = 6$, $f(A,B,C,D,E,F)$, is shown below, with solid lines indicating the two 5-variable subarrays and dotted lines indicating the four 4-variable sub-subarrays.

	U	U	U	U	U	U	U	U	U	U	U	U	U	U
$\overline{U}$	$\overline{A}$	$\overline{A}$	$\overline{A}$	$\overline{A}$	$\overline{A}$	$\overline{A}$	$\overline{A}$	$\overline{B}$	$\overline{C}$	$\overline{C}$	$\overline{C}$	$\overline{D}$	g	$\overline{E}$
$\overline{U}$	$\overline{B}$	$\overline{C}$	$\overline{C}$	$\overline{C}$	$\overline{D}$	g	$\overline{E}$	$\overline{B}$	$\overline{D}$	g	$\overline{E}$	g	E	g
$\overline{U}$	$\overline{B}$	$\overline{D}$	g	$\overline{E}$	g	E	g	$\overline{B}$	g	E	g	$\overline{E}$	g	D
$\overline{U}$	$\overline{B}$	g	E	g	$\overline{E}$	g	D	$\overline{B}$	$\overline{E}$	g	D	C	C	C
$\overline{U}$	$\overline{B}$	$\overline{E}$	g	D	C	C	C	$\overline{C}$	$\overline{C}$	$\overline{C}$	D	g	$\overline{E}$	B
$\overline{V}$	$\overline{C}$	$\overline{C}$	$\overline{C}$	$\overline{D}$	g	$\overline{E}$	B	$\overline{D}$	g	$\overline{E}$	g	E	g	B
$\overline{V}$	$\overline{D}$	g	$\overline{E}$	g	E	g	B	g	E	g	$\overline{E}$	g	D	B
$\overline{V}$	g	E	g	$\overline{E}$	g	D	B	$\overline{E}$	g	D	C	C	C	B
$\overline{V}$	$\overline{E}$	g	D	C	C	C	B	A	A	A	A	A	A	A

where the 32 nonfixed inputs g_{00000} through g_{11111} are all denoted simply g.

The factorization array, unless it has prewired fixed inputs, can often be reduced for a given function. The factorization array factors a function into subfunctions, each of which is in turn factored until eventually each sub-subfunction consists of a single literal. Each subfunction is realized in a $\overline{U}U$ SBC subarray. Many of the subfunctions may have $\overline{U}U$ SBC MAJA realizations smaller than the one used in the factorization array. These can be substituted for the standard subarray with a corresponding decrease in array size.

EMBEDDED SUBARRAYS

In this section the process of embedding subarrays will be considered in general. If one has two $\overline{U}U$ SBC MAJA's realizing two self-dual functions

$$g_0^{sd} = Ug_0 + \overline{U}g_0^d$$

and

$$g_1^{sd} = Ug_1 + \overline{U}g_1^d$$

then these two arrays can be embedded in a WV SBC MAJA to realize the self-dual function

$$f^{sd} = VXg_0 + VYg_1 + WXg_1^d + WYg_0^d + VW + WXY$$

where X and Y are any single literals. If the array for g_0^{sd} is denoted

	U	$U \ldots U$
$\overline{U}$	δ	$\delta \ldots \delta$
$\overline{U}$	δ	δ .
.	.	.
.	.	.
.	.	.
$\overline{U}$	δ	$\delta \ldots \delta$

where δ denotes the various inputs of the array for g_0^{sd}, and if the array for g_1^{sd} is denoted, similarly,

	U	$U \ldots U$
$\overline{U}$	ϵ	$\epsilon \ldots \epsilon$
$\overline{U}$	ϵ	ϵ .
.	.	.
.	.	.
.	.	.
$\overline{U}$	ϵ	$\epsilon \ldots \epsilon$

then the array for f^{sd} is

	V	$V \ldots .$	.	$\ldots V$
W	X	X	ϵ	$\epsilon \ldots \epsilon$
W	δ	δ	ϵ	ϵ .
.	δ	δ .		.
.	.	.		.
.	.	$\ldots .$		.
.	.	$\ldots .$	ϵ	$\epsilon \ldots \epsilon$
W	δ	$\delta \ldots \delta$	Y	$Y \ldots Y$

Array 8

In this illustration the arrays for g_0^{sd} and g_1^{sd} have been assumed to be of equal height. This need not be true in general.

The formal definition of this construction follows.

Given a self-dual function $f^{sd}(x_1,x_2,\ldots,x_n)$, express it in a VW factorization as $f^{sd} = Vg + Wg^d + VW$ where the function g may or may not be self-dual. This is always possible, if V and W are properly chosen from $\{x_1,\bar{x}_1,x_2,\ldots,\bar{x}_n\}$. Then factor g as $g = Xg_0 + Yg_1$. Again, this is always possible.

Then

$$g^d = Xg_1^d + Yg_0^d + XY$$

and

$$f^{sd} = VXg_0 + VYg_1 + WXg_1^d + WYg_0^d + VW + WXY$$

with $V,W,X,$ and Y chosen from $\{x_1,\bar{x}_1,x_2,\ldots,\bar{x}_n\}$.

Now construct an SBC UU MAJA to realize $g_0^{sd} = Ug_0 + Ug_0^d$. Call this array A_0. Let its size be $h_0 \times w_0$. Similarly construct the SBC UU MAJA A_1 of size $h_1 \times w_1$ to realize $g_1^{sd} = Ug_1 + Ug_1^d$. Note that there is no restriction on how A_0 and A_1 are constructed, or on their size. It will now be shown that the two arrays A_0 and A_1 can be embedded in an SBC WV MAJA, A, of size $h \times (w_0 + w_1)$ which realizes f^{sd}, where h equals the larger of $h_0 + 1$ and $h_1 + 1$.

Let the center input to element ij of A_0 be called α^0_{ij}, defined for all i, j: $1 \leq i \leq h_0$, $1 \leq j \leq w_0$. Similarly, let the center input to element ij of A_1 be called α^1_{ij}, defined for all i, j: $1 \leq i \leq h_1$, $1 \leq j \leq w_1$. Then the inputs to the $h \times w$ array A are assigned as follows, where h = the largest of $h_0 + 1$ and $h_1 + 1$ and $w = w_0 + w_1$ and α_{ij} denotes the center input to element ij of A:

For $1 \leq i \leq h - h_0$ and $1 \leq j \leq w_0$
$\alpha_{ij} = X$

For $h - h_0 < i \leq h$ and $1 \leq j \leq w_0$
$\alpha_{ij} = \alpha^0_{kj}$ with $k = i - (h - h_0)$

For $1 \leq i \leq h_1$ and $w_0 < j \leq w$
$\alpha_{ij} = \alpha^1_{i(j-w_0)}$

For $h_1 < i \leq h$ and $w_0 < j \leq w$
$\alpha_{ij} = Y$

This specifies every input to A in terms of X and Y and the inputs to A_0 and A_1. Array 8 is an example. It can be proved[1] that the array just defined has as output the function $f^{sd}(x_1,\ldots,x_n)$.

It is very important to observe that the only restrictions on the arrays A_0 and A_1 are

1. That they are SBC arrays with U and $\bar{U}$ as boundary variables.
2. That they realize $g_0^{sd} = Ug_0 + Ug_0^d$ and $g_1^{sd} = Ug_1 + Ug_1^d$ respectively.

Condition (2) is *not* equivalent to the condition (2′): That when $U = 1$ and $U = 0$, A_0 and A_1 realize g_0 and g_1 respectively.

Since the subarrays A_0 and A_1 can be any UU SBC MAJA's realizing the functions g_0^{sd} and g_1^{sd} respectively, it is possible to construct one or both of A_0 and A_1 themselves as factored arrays. In fact, the canonical factorization array for n variables is just a factored array with each subarray factored and each sub-subarray factored and so on until each sub-sub . . . subarray is a 3×3 canonical array which realizes a function of only three variables.

To illustrate the use of factored subarrays in a factored array in the general case, express g_0^{sd} and g_1^{sd} as

$$g_0^{sd} = UR_0g_{00} + US_0g_{01} + UR_0g_{01}^d + US_0g_{00}^d + UR_0S_0$$

and

$$g_1^{sd} = UR_1g_{10} + US_1g_{11} + UR_1g_{11}^d + US_1g_{10}^d + UR_1S_1$$

and realize each of them in factored UU arrays, which are used as subarrays in the array for f^{sd}. Figure 4 shows the construction of the resulting array. In this array the function f^{sd} has been factored as

$$f^{sd} = VXR_0g_{00} + VXS_0g_{01} + VYR_1g_{10} + VSY_1g_{11} + VYR_1S_1 + WXR_1g + WYR_0g_{01} + WYS_0g_{00} + WXY + VW$$

(For the sake of illustration it has been assumed in Fig. 4 that g_{00}^{sd} can be realized in a 2×2 SBC UU MAJA, while g_{01}^{sd}, g_{10}^{sd}, and g_{11}^{sd}, each require a 3×3 SBC UU MAJA.)

A study has been made of two-dimensional arrays of three-input one-output gates, or elements, each element realizing the majority function of its three inputs ($f(A,B,C) = AB+AC+BC$). These arrays are functionally equivalent to arrays of minority elements ($f(A,B,C) = AB+AC+BC$).

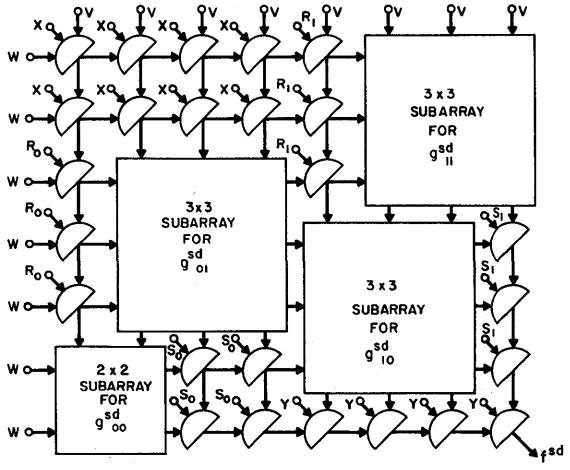

Figure 4. Four subarrays embedded in an SBC factorization MAJA.

SUMMARY

Two methods are developed for synthesizing any given Boolean function in an array. The first method results in an array whose size depends on the particular function being realized. The second method results in an array whose size depends only on the number of variables in the function being realized. Any 4-variable function, for example, can be realized in an array of 24 elements or less.

The principle result of this work is a simple algorithmic synthesis procedure with the following properties:

1. It is based on building blocks (arrays) which are characterized solely by their width and height, and which contain only simple three-input, one-output elements of *one* type, with a maximum output load of two elements each.
2. It results in arrays obeying a known upper bound on size that seems reasonably small.
3. It permits the synthesis of any Boolean function of n-variables by specifying no more than $2^{(n-1)}$ inputs to the array.
4. It permits the logical decomposition of the array into subarrays, corresponding to a decomposition of the function into subfunctions, with no physical modification of the array.
5. It results in circuits (arrays) with a longer delay, and hence lower speed, than conventional logic circuits.
6. It often requires more elements to realize a given function than do methods less constrained in element type and interconnection.

REFERENCES

1. R. H. Canaday, "Two-Dimensional Iterative Logic," Report ESL-R-210 Electronic Systems Laboratory, Massachusetts Institute of Technology, Cambridge, Mass., (Sept. 1964). The same material appears in: R. H. Canaday, "Two-Dimensional Iterative Logic," M.I.T. Department of Electrical Engineering, Ph.D. Thesis, Sept. 1964.

2. S. B. Akers, Jr., "The Synthesis of Combinational Logic Using Three-Input Majority Gates"; *Third Annual Symposium on Switching Circuit Theory and Logical Design,* Chicago, October 7-12, 1962.

3. S. H. Caldwell, *Switching Circuits and Logical Design,* Wiley and Sons, New York, 1958.