

Universal Logic Blocks

BARRY P. SHAY

*Information Systems Branch
Mathematics and Information Sciences Division*

September 17, 1969



NAVAL RESEARCH LABORATORY
Washington, D.C.

CONTENTS

Abstract	ii
Problem Status	ii
Authorization	ii
INTRODUCTION	1
THE UNIVERSAL LOGIC BLOCK	1
DISCOVERING UNIVERSAL LOGIC FUNCTIONS	6
DESIGNING WITH UNIVERSAL LOGIC BLOCKS	11
PROPOSED INVESTIGATION	13
REFERENCES	15

ABSTRACT

Combinatorial networks are usually synthesized from integrated-circuit building blocks. Advances in integrated-circuit processing technology permit the placement of more logic on a single block, but raise two new questions as to what logical functions should be placed on a block and how combinatorial networks can be efficiently synthesized from a family of blocks.

The Universal Logic Block (ULB) provides a promising approach towards answering these questions. A ULB produces a particular function of n Boolean variables (and its complement) that has the property of covering any Boolean function of k or fewer variables by suitable input partitioning. Such functions are known as Globally Universal (GU) functions.

Currently available methods of finding GU functions depend upon realizing certain equivalence classes of Boolean functions. The enormous growth rate of these classes limit the practicality of such methods. Techniques must be found which either avoid the realization of equivalence classes, or enlarge each class by defining new relations on the set of Boolean functions.

The synthesis of large networks using the GU block as the logical connective remains a difficult problem. By placing constraints on the desired network, realistic design criteria can be established. Considerations to be taken into account include size of the block to be used, delays associated with the block, and the mixing of various types of blocks.

PROBLEM STATUS

This is an interim report on one phase of the problem; work on this and other phases is continuing.

AUTHORIZATION

NRL Problem B01-04
Project RR 003-02-41-6150

Manuscript submitted April 25, 1969.

UNIVERSAL LOGIC BLOCKS

INTRODUCTION

It is increasingly evident that conventional techniques for the design of complex digital equipment are becoming obsolete (1). The standard minimization procedures and the corresponding reduced cost functions are now, or will be in the future, no longer applicable. The current state of the art in the semiconductor fabrication industry demands that new design techniques use constructively the inherent features of this new technology.

In the past, logical design was based upon the optimum use of logic gates (NAND, AND, etc.) to realize given switching functions. It was desired to minimize the number of diodes (input leads) and/or transistors used. Using integrated circuits however, logical modules can be produced at a cost practically independent of the number of diodes and transistors contained in the module. Therefore it seems reasonable to assume that the cost of a complex switching circuit composed of such modules will be determined by the number of modules used, together with the number of interconnections between modules, and not by the complexity of circuits within a module. The problem then is two-fold. First, efficient design techniques must be established for the synthesis of digital circuits using a fairly complicated module as the logical connective. Second, logical modules must be found which simplify the synthesis procedure while decreasing the associated cost.

Relatively little work has been done in the past concerning these two problems. The most ambitious attempt on the former has been done by Patt (2) and Schneider (3). Patt defines a Well Ordered Sequence (WOS) module and develops a minimal two-level synthesis procedure using this module. Schneider derives an algorithm for the synthesis of combinatorial circuits from an arbitrary collection of integrated circuits. More will be said about both works later.

Presently the choice of an efficient building block is an open question. Early efforts in this aspect were made by Forslund and Waxman (4), Dunham and North (5), and Dunham, et al. (6). More recently Elspas et al. (7), Susskind, et al. (8) and Yau and Tang (9) have, contributed by dealing with "Globally Universal Functions" (GUF's). Of these, however, only Yau attempted to synthesize larger networks from smaller universal modules. As will be seen, his approach, although interesting, is limited in nature.

THE UNIVERSAL LOGIC BLOCK

A candidate for a functional block used in the synthesis of larger networks is the Universal Logic Block (ULB), also referred to as the Globally Universal Block (GUB). It has been shown by Susskind (8) that networks can be designed using GUB's with reduced cost functions, even if no formal design procedure is used.

Consider the following cost criteria: a given circuit configuration is scored by adding the total number of signal-level ties which must be made between integrated-circuit modules to the total number of input and output signal lines. Using GUB's circuits can be obtained having scores significantly lower than circuits obtained using the standard minimization techniques of Quine-McClusky.

Obviously a cost criteria that must be considered in any large scale design is the number of modules used. This problem will be discussed in the following two sections of this report.

We begin our discussions of GUF's by presenting some basic definitions.

1. Let F be a Boolean function of k variables. Let $(k - j)$ of the variables be eliminated by setting them equal to constants. A function G of the remaining j variables is thus obtained. Function G is said to be a $(k - j)^{th}$ order subfunction of F obtained by biasing.

2. Let $(k - j)$ of the variables be eliminated by setting them equal to other variables. A function of j variables is obtained. Function G is said to be a $(k - j)^{th}$ order subfunction of F obtained by duplication.

3. Two functions, $f(x_1, \dots, x_k)$ and $g(y_1, \dots, y_k)$, are said to be equivalent under permutation if and only if there exists some one-to-one mapping

$$M: X \rightarrow Y$$

such that

$$f[M(x_1), \dots, M(x_k)] = g(y_1, \dots, y_k).$$

4. Two functions, $f(x_1, x_2, \dots, x_k)$ and $g(y_1, \dots, y_k)$, are said to be equivalent under permutation and complementation if and only if there exists some one-to-one mapping

$$M: X^* \rightarrow Y$$

such that

$$f[M(x_1^*), \dots, M(x_k^*)] = g(y_1, \dots, y_k)$$

where

$$x_i^* = \begin{cases} x_i & \text{or} \\ \bar{x}_i \end{cases}$$

5. Let f be a function of k variables. The set consisting of f and all functions equivalent to f under some (permutation and/or complementation) operation is called an equivalence class under that operation.

6. Let f be a function of k variables. The function f is said to be globally universal in $j < k$ variables if and only if the set of $(k - j)^{th}$ order subfunctions of f obtained by both biasing and duplication contain at least one member of every permutation equivalence class of j variable functions, or the complement of a member.

Therefore we have the following: Given that $f(x_1, \dots, x_k)$ is globally universal in $j < k$ variables, there exists some way of biasing and/or duplicating input leads so that a member of any desired permutation equivalence class of j variables (or its complement) can be realized. Since the j input variables may be attached to the input leads in any order, every member of each permutation equivalence class of j variables may be obtained. Thus a globally universal j block (GU j) realizes every function of j variables. Also it can be shown that if $f(x_1, \dots, x_k)$ is GU j , then so is $f(x_1, \dots, x_k)$, $f(\bar{x}_1, \dots, \bar{x}_k)$ and $f(\bar{x}_1, \dots, \bar{x}_k)$.

To be more explicit, consider the following example. There are ten non-trivial functions of two variables, and they are listed below in groups according to the equivalence class, to which they belong. All functions with the same designation are members of the same permutation equivalence class.

- | | |
|-------------------------|-------------------------------|
| 1. AB | 5. $\bar{A} + \bar{B}$ |
| 2. $\bar{A}B, A\bar{B}$ | 6. $\bar{A} + B, A + \bar{B}$ |
| 3. $\bar{A}\bar{B}$ | 7. $\bar{A}B + A\bar{B}$ |
| 4. $A + B$ | 8. $\bar{A}\bar{B} + AB$ |

It is seen in the following equations how members of the same equivalence class are obtained by merely permuting the input variables.

$$1. AB \Big]_{\substack{A \rightarrow B \\ B \rightarrow A}} = BA = AB$$

$$2. \bar{A}B \Big]_{\substack{A \rightarrow B \\ B \rightarrow A}} = \bar{B}A = A\bar{B}$$

$$3. \bar{A}\bar{B} \Big]_{\substack{A \rightarrow B \\ B \rightarrow A}} = \bar{B}\bar{A} = \bar{A}\bar{B}$$

$$6. \bar{A} + B \Big]_{\substack{A \rightarrow B \\ B \rightarrow A}} = \bar{B} + A = A + \bar{B}$$

$$7. \bar{A}B + A\bar{B} \Big]_{\substack{A \rightarrow B \\ B \rightarrow A}} = \bar{B}A + B\bar{A} = \bar{A}B + A\bar{B}$$

Now we can test the function $f(x_1, x_2, x_3) = \bar{x}_1x_2 + x_1\bar{x}_3$ to see whether it is globally universal in two variables (i.e., can we realize at least one member of each equivalence class, or its complement).

By biasing we have:

$$f(A, 1, B) = \bar{A} + A\bar{B} = \bar{A} + B \implies 5, \bar{1}$$

$$f(A, B, 1) = \bar{A}B, \bar{f}(B, A, 1) = A\bar{B} \implies 2, \bar{6}$$

$$f(A, B, 0) = \bar{A}B + A = A + B \implies 4, \bar{3}$$

By duplication we have:

$$f(A, B, B) = \bar{A}B + A\bar{B} \implies 7, \bar{8}$$

Therefore according to the definition of GU 2 functions, we see that $f(x_1, x_2, x_3)$ is such a function.

Notice that we have only used four variations of $f(x_1, x_2, x_3)$ to realize at least one member of each equivalence class or its complement. Similarly it can be verified that the function $g(x_1, x_2, x_3) = x_1x_2 \oplus x_3$ is globally universal in two variables.

There are 68 equivalence classes of three variables under permutation. For a function to be GU 3, it and its complement must be able to realize at least one member of each equivalence class, by partitioning the inputs in some way. It can be shown that no four-variable function can be GU 3 (i.e., the number of three-variable subfunctions

is 28). Therefore we can not realize all three-variable functions with a single four-variable function. There do exist, however, many five-variable functions which are GU 3. For example, the function

$$f(A,B,C,D,E) = \bar{A}CD + \bar{B}C\bar{D}\bar{E} + AB\bar{D}\bar{E} + \bar{B}C\bar{D}\bar{E} + BC\bar{D}E + ABC\bar{D}\bar{E} + \bar{B}C\bar{D}\bar{E}$$

can be shown to be GU 3. It should be noted that only true inputs are required at all times.

At this point it is interesting to compare the preceding with similar works of Elspas, et al. (7). In Susskind's work it is seen that in reality only equivalence under permutation is considered, although equivalence under permutation and complementation is mentioned. Thus it is assumed that only true variables are available at the input. Also, since the basic ULM will have complementary outputs, each time a function is realized we can assume its complement is also realized. Thus the 68 equivalence classes of three variables can be broken down into ten genera. Two functions f and g are said to be of the same genera if and only if f and g are equivalent under permutation and complementation or f and \bar{g} are equivalent under permutation and complementation. Using these concepts, Elspas presents upper and lower bounds on the minimum number of input leads necessary for a block to be universal. Susskind presents only lower bounds based upon equivalence under permutation alone, and the fact that half of the permutation equivalence classes are realized by the GU function and the other half by its complement.

The upper bound is derived by an iterative construction involving low-order cases. In the process, Elspas also shows that the existence of a four-input GU 3 block is impossible.

We have already exhibited a GU 3 block with five inputs; thus the minimum number of inputs required for a GU 3 block is exactly five. The best upper and lower bounds are given in Table 1.

Table 1
Minimum Upper and Lower Bounds on the Number
of Inputs a Block Must Have to be GUJ

J	Susskind Lower Bound	Elspas	
		Best Lower Bound	Best Upper Bound
2	3	3	3
3	5	4	5
4	7	6	8
5	12	10	18
6	22	17	37

Patt (2), in agreeing that the cost of a switching network depends on the number of modules used, considers the problem of selecting and using a "building block" module. Although universality of the module is not his goal, the larger problem of synthesizing any network is discussed. Since it is desired to use modules all of a single type, they must be complete. That is, any desired function must be realizable using only that module as the logical connective. Aside from this property, the block should be asymmetric and versatile. A switching function f is said to be totally asymmetric if every permutation of its input variables results in a different output function. Therefore it can realize $n!$ different functions of n variables. A function f of k variables is a

subfunction of g of $n > k$ variables, if f can be obtained from g by biasing and/or duplicating one or more of the input variables of g . Thus the more versatile a module is, the greater the variation in the subfunction of the module. Under this definition, it is clear that a GUB would be considered most versatile. Patt (2) introduces the WOS module as one that is complete, asymmetric, and logically versatile. It is a module of n variables (n - inputs) and one output which realizes the following switching functions.

$$f = 1 \oplus x_1 \oplus x_3 \oplus x_4 \oplus \dots \oplus x_n \oplus x_1 x_2 \oplus x_1 x_2 x_3 \oplus \dots \oplus x_1 x_2 \dots x_{n-1}$$

A procedure which leads to a two-level realization of any function is developed using a minimum number of WOS modules. It is pointed out that no optimal synthesis procedure exists for the three-variable WOS module, but that by following a few guidelines low-cost networks can be designed. More will be said about this subject later in this report.

Yau and Tang (9), without resorting to the use of equivalence classes, presents a universal block in a direct manner. Note that any logical function of three variables can be expanded as follows:

$$f(x_1, x_2, x_3) = \bar{x}_1 \bar{x}_2 f(0, 0, x_3) + \bar{x}_1 x_2 f(0, 1, x_3) + x_1 \bar{x}_2 f(1, 0, x_3) + x_1 x_2 f(1, 1, x_3)$$

Now $f(0, 0, x_3)$, $f(0, 1, x_3)$, $f(1, 0, x_3)$ and $f(1, 1, x_3)$ are all functions of x_3 only, and each of these functions assumes one of the four values: x_3 , \bar{x}_3 , 0, or 1. Thus the circuit of Fig. 1 can realize this function.

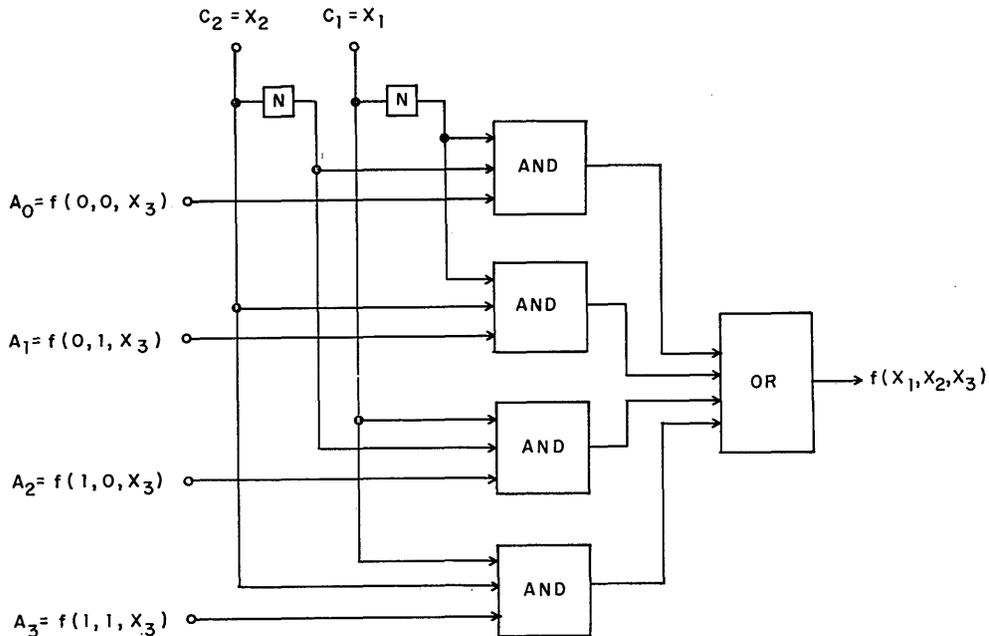


Fig. 1 - Realization of a general three-variable Boolean function

A slight advantage of this circuit configuration is that biasing to 0 or 1 is not necessary. For example, if $f(0, 1, x_3) = 1$, then tying A_1 to x_2 would yield the same result. Similarly, if $f(0, 1, x_3) = 0$, then tying A_1 to x_1 would yield the same result.

Consider now a function of four variables, and expand it as follows:

$$f(x_1, x_2, x_3, x_4) = \bar{x}_1 \bar{x}_2 \bar{x}_3 f(0, 0, 0, x_4) + \bar{x}_1 \bar{x}_2 x_3 f(0, 0, 1, x_4) \\ + \bar{x}_1 x_2 \bar{x}_3 f(0, 1, 0, x_4) + \dots + x_1 x_2 x_3 f(1, 1, 1, x_4)$$

The corresponding logic circuit is shown in Fig. 2.

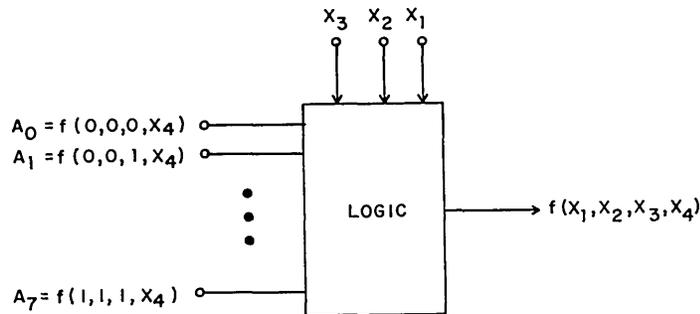


Fig. 2 - Universal four-variable function

In a like manner, ULB's in five or more variables can be expanded and then represented by a similar configuration. A design procedure using ULB's in three-variables to realize function of n variables will be discussed later.

DISCOVERING UNIVERSAL LOGIC FUNCTIONS

As has been mentioned before, one of the major problems is determining globally universal functions of n variables. Yau's procedure, which will be discussed later, requires $2^{n-1} + (n-1)$ input terminals, which for even reasonable values of n is an enormous number. However, Yau avoids the problem of realizing extremely large numbers of equivalence classes, and in this sense his solution is tractable.

Since our main objective is to realize large networks composed of smaller building blocks, the remainder of this section will deal with finding universal functions in two, three, and four variables. To begin with, we will consider a direct algebraic approach, which by its nature is useful only in finding GU 2 functions.

Any two-variable Boolean function can be expanded in the canonical AND-OR form, as follows:

$$f(x_1, x_2) = a_0 \bar{x}_1 \bar{x}_2 + a_1 \bar{x}_1 x_2 + a_2 x_1 \bar{x}_2 + a_3 x_1 x_2$$

where the values (0,1), for each coefficient a_i , determines the exact function. We will consider only the nondegenerate functions of exactly two variables. The ten possible combinations of the a_i 's which give rise to the ten functions are shown in Table 2. To the right of these functions is the corresponding permutation equivalence class, which the function is a member.

Table 2
Generation of the Ten Nondegenerate Functions of Two Variables
and their Equivalence Classes.

Coefficients				Function	Equivalence Class	Complement of Equivalence Class
a_0	a_1	a_2	a_3			
0	0	0	1	$x_1 x_2$	1	5
1	1	1	0	$\bar{x}_1 + \bar{x}_2$	5	1
0	0	1	0	$x_1 \bar{x}_2$	2	6
0	1	0	0	$\bar{x}_1 x_2$	2	6
1	1	0	1	$\bar{x}_1 + x_2$	6	2
1	0	1	1	$x_1 + \bar{x}_2$	6	2
1	0	0	0	$\bar{x}_1 \bar{x}_2$	3	4
0	1	1	1	$x_1 + x_2$	4	3
0	1	1	0	$\bar{x}_1 x_2 + x_1 \bar{x}_2$	7	8
1	0	0	1	$\bar{x}_1 x_2 + x_1 x_2$	8	7

Now we must realize each permutation equivalence class, or its complement, by modifying a general three-variable function in some way. Any three-variable function may be expressed as:

$$f(x_1, x_2, x_3) = b_0 \bar{x}_1 \bar{x}_2 \bar{x}_3 + b_1 \bar{x}_1 \bar{x}_2 x_3 + \dots + b_7 x_1 x_2 x_3.$$

We now take all nine first-order subfunctions and impose conditions on the b_j 's such that at least one member of each equivalence class of two variables is realized. The nine subfunctions are listed as follows:

<u>Biasing</u>	<u>Duplication</u>
$x_1 = 0, 1$	$x_1 = x_2$
$x_2 = 0, 1$	$x_1 = x_3$
$x_3 = 0, 1$	$x_2 = x_3$

For example, if we set $x_1 = 0$ we obtain

$$f(0, x_2, x_3) = b_0 x_2 x_3 + b_1 \bar{x}_2 x_3 + b_2 x_2 \bar{x}_3 + b_3 x_2 x_3.$$

For $x_2 = 0$, we have

$$f(x_1, 0, x_3) = b_0 \bar{x}_1 \bar{x}_3 + b_1 \bar{x}_1 x_3 + b_4 x_1 \bar{x}_3 + b_5 x_1 x_3.$$

In a similar manner we obtain all nine subfunctions of $f(x_1, x_2, x_3)$.

Now suppose we want to realize a function from equivalence class 1 or its complement (i.e., a function of the form $x_i x_j$). It can be seen that for a given equivalence class or its complement to be realized by a subfunction, one or more sets of the b coefficients must be equal to the b coefficients for that class. For the $x_1 = 0$ subfunction, setting $b_0 = b_1 = b_2 = 0$, and $b_3 = 1$, gives us $x_2 x_3$, and setting $b_0 = b_1 = b_2 = 1$ and $b_3 = 0$ gives us $\bar{x}_2 + x_3$. Thus to realize class 1 or its complement we have the condition, expressed logically,

$$\bar{b}_0 \bar{b}_1 \bar{b}_2 b_3 + b_0 b_1 b_2 \bar{b}_3 + 1.$$

Similarly we can realize class 1 or its complement from each of the other eight subfunctions, so that finally we have the following equation containing nine terms:

$$(\bar{b}_0 \bar{b}_1 \bar{b}_2 b_3 + b_0 b_1 b_2 \bar{b}_3) + (\bar{b}_0 \bar{b}_1 \bar{b}_4 b_5 + b_0 b_1 b_4 \bar{b}_5) \\ + \dots + (\bar{b}_0 \bar{b}_2 \bar{b}_4 b_7 + b_0 b_3 b_4 \bar{b}_7) = 1,$$

where the first parenthesizal term corresponds to subfunction 1, the second to subfunction 2, and the last to subfunction 9. A similar equation can now be written for each of the three remaining equivalence classes, giving us a total of four equations in eight unknowns (the b_i 's). Since there is no direct method of solving a set of Boolean simultaneous equations, a rather involved procedure yields the following five functions which are globally universal in two variables.

1. $\bar{x}_1 \bar{x}_2 x_3 + x_1 x_2 x_3$
2. $x_1 x_3 + x_1 \bar{x}_2 + \bar{x}_1 x_2 \bar{x}_3$
3. $\bar{x}_1 \bar{x}_2 + x_1 \bar{x}_3$
4. $\bar{x}_1 \bar{x}_2 + x_1 x_2 \bar{x}_3$
5. $\bar{x}_1 \bar{x}_2 + x_1 x_2 \bar{x}_3$

It should be noted that each of these functions falls into a distinct permutation equivalence class of three variables.

The method presented above is unsatisfactory for determining GU functions of more than two variables. For the three-variable case, 34 equations of 32 variables each would arise.

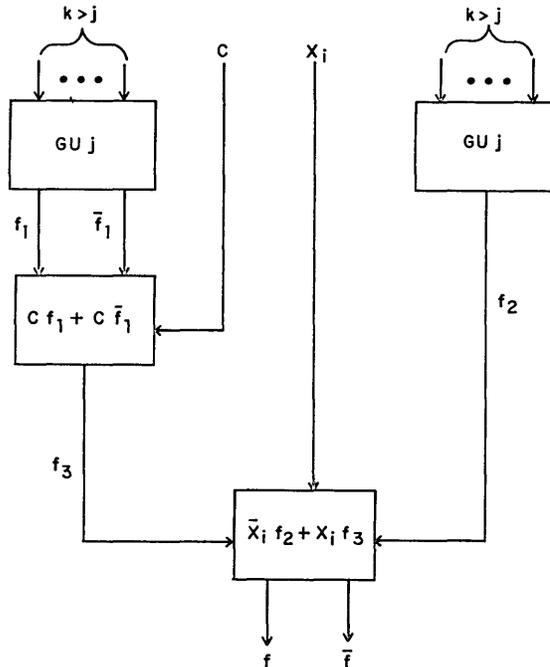
Elsplas (7), by considering the nondegenerate functions of two variables composed of two general exhibits essentially the same functions as does Susskind for GU 2 blocks. Since complements are allowed at the inputs, the two genera are $x_1 x_2$ and $x_1 \oplus x_2$. It is easy to see, then, that a function of the form $x_1 x_2 \oplus x_3$ is universal in two variables. In addition, one need only modify x_2 and/or x_3 to realize all two variable functions.

Both Elspas and Susskind use a similar method to arrive at a GU $(j + 1)$ block from two GU j blocks, by decomposing a function of $n + 1$ variables into two functions of n variables. This procedure will be briefly described now. Any function of $j + 1$ variables can be expanded about any variable as follows:

$$f(x_1, \dots, x_i, \dots, x_{j+1}) = \bar{x}_i f_2(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_{j+1}) \\ + x_i f_3(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_{j+1}).$$

Now f_2 and f_3 can each be realized by GU j blocks, as shown in Fig. 3. It should be noted that in the decomposition, provision must be made for selecting either f_1 or \bar{f}_1 , depending upon whether f_2 is f_2 or in fact is \bar{f}_2 . This is done by the line labeled C . That is, if $C = 1$, then $f_3 = f_1$, and if $C = 0$, then $f_3 = \bar{f}_1$. It should also be noted that this configuration is wasteful of inputs. Whereas each GU j block required k input leads, the GU $(j + 1)$ block requires $2k + 2 = 2(k + 1)$ input leads (i.e., for increasing the universality by one, we more than double the number of input leads). For $j = 2$, then, we require eight input leads to realize a GU 3 block, whereas it was shown that we only need five input lines. The advantage of this procedure becomes apparent when larger values of j are required; in addition, no complicated table of input partitions is needed to realize a specific function.

Fig. 3 - Circuit realization of expanded function



We now look at methods for determining functions which are universal in three and four variables. The techniques are in general searches over k variable functions or the classes to which they belong. Since for $j > 5$ the number of functions and equivalence classes (and genera) are enormous, it seems reasonable to restrict the universe to be searched in some way. Elspas (7) does this by (a) considering genera which are most nearly universal, and (b) considering functions to be representatives of "families."

By exhaustive search methods it was found that the following three functions of four variables realize nine of the ten genera of three variables.

- a. $wxz + wyz + xy\bar{z} + \bar{w}\bar{x}\bar{y}\bar{z}$
- b. $wxy + wx\bar{z} + wyz + \bar{w}\bar{x}\bar{y}\bar{z} + \bar{w}\bar{x}y\bar{z}$
- c. $wxz + wy\bar{z} + \bar{w}\bar{x}\bar{y}\bar{z} + \bar{w}\bar{x}y\bar{z}$

To construct a GU 3 block from these four input blocks, a fifth input is introduced as shown in Fig. 4.

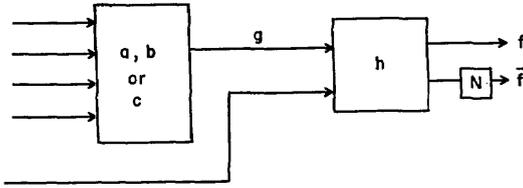


Fig. 4 - Construction of a GU 3 block

When a is used, then h is an inclusive OR block, whereas when b or c is used, h should be an exclusive OR block. The tenth genus is produced by tying the fifth input to a variable and partitioning the other four inputs such that $g = y + z$ if C is used, and $g = y \oplus z$ if a or b is used.

Susskind has also determined five input functions which are GU in three variables using a computer search method. Whereas Elspas (7) searched first for four variable functions which are almost universal, Susskind proceeds directly to search for five-variable GU 3 functions. Basically the system generates a 32-bit random number, which then represents an initial five-variable function. A program then generates the 125 three-variable subfunctions, and determines the number of three-variable permutation equivalence classes found among them. If this number is 68, then the initial function is GU 3. If this number is less than 68, the initial function is modified in an attempt to increase the number of equivalence classes found. The procedure includes an equivalence-checking program which determines whether or not a GU 3 function has already been found. In addition, the programming system determines the number of five-variable equivalence classes which are globally universal in three variables, and the number of three-variable equivalence classes realizable by any five-variable function under test, and it tests hypotheses concerning GU functions. A similar system for GU 4 functions would require about two man-months to implement; one for GU 5 function is impractical at the present time.

As previously mentioned, Elspas, in an attempt to further restrict classes to be searched, introduces the concept of a family of functions. Two functions, f and g , are said to be of the same family if and only if there exists a linear function of input variables $l(x_1, \dots, x_n)$ such that f and $g \oplus l$ are of the same genus. Thus all functions of the same genus are of the same family (i.e., let $l = 0$). The subsequent search method begins by selecting an m -variable test function and classifying all of its subfunctions by genus. The final network consists of a block realizing the test function followed by r_{\max} exclusive OR blocks, and thus has $m + r_{\max}$ total inputs. The r_{\max} exclusive OR blocks essentially represent the linear function $l(x_1, \dots, x_n)$; r_{\max} is defined as

$$r_{\max} = \max_i [r_i],$$

where r_i is the maximum number of exclusive OR modules required to realize the entire i^{th} family, given that the test function can produce one or more genera of the i^{th} family for all i .

The actual eight-input GU 4 block realized was found by using a six-variable test function, which realizes 214 of the 222 four-variable genera using only a single exclusive OR at the output.

King (10) has constrained the problem of finding a universal block to that of finding a block which realizes a specified set of functions. His solution is extremely worthwhile, since it presents a clear-cut algorithmic solution to this related problem. Specifically he has solved the following. Given a set of v functions (f_1, f_2, \dots, f_v) , find a logic block and accompanying partition of input variables that will realize each function in the set.

DESIGNING WITH UNIVERSAL LOGIC BLOCKS

Presently no systematic design procedure exists for the synthesis of networks using globally universal blocks as the basic logical connective. This deficiency is illustrated by the following procedure, mentioned in Susskind.

1. Obtain the minimum sum-of-products expression for the desired function.
2. Draw the standard AND-OR network for the function.
3. Assuming that GU j blocks are available for k inputs, with k a constant, circle portions of the network involving j or fewer variables.
4. For each circled portion of the network, substitute a GU j block if the lead count would be reduced by so doing.

It should be noted that an approach similar to the above, although not dealing with universal blocks, has been developed for the IBM System 360 (11). In this approach, standard techniques are used for minimization which are independent of logical blocks. The completed design is now examined by computer. Adjacent clusters of logic are thereby matched with groupings of logic found on functional blocks. These groupings are then assigned to appropriate cards. The method described is very fast, but not sophisticated.

The following example will illustrate the former method. Suppose

$$f(A, B, C) = ABC + \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} = A \oplus B \oplus C.$$

To realize this supposition, we could use a globally universal 3 block, with five-input leads. The lead count for such a block is 7, including the unused complementary output, whereas standard AND-OR circuitry would require 21 leads.

It should be noted that the saving involved is not only in lead count. If it is assumed that the manufacturer has GU 3 blocks on the shelf, then in effect, he already has designed the circuit. All that is then needed is the proper partitioning of the inputs to the block. It should also be noted, however, that partitioning the inputs to obtain a particular function is not a solved problem. Presently, a table would accompany a GU j block for each j , specifying the proper partition for a desired functional class. It is an easy matter to obtain the function once the equivalence class is realized.

Many logical functions have obvious decompositions (12). That is, a Boolean function may be expressed as a function of other functions. For example, if

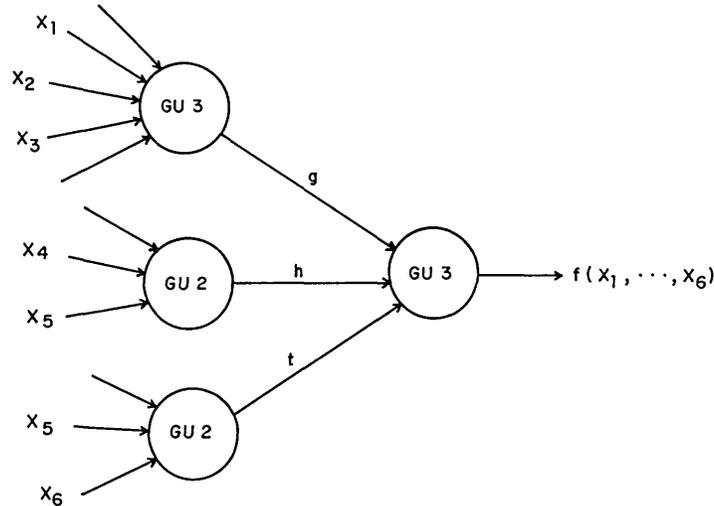
$$f_1(x_1, x_2, \dots, x_3) = f_2 \left[g(x_1, x_2, x_3), h(x_4, x_5), t(x_5, x_6) \right]$$

then it could be realized by two GU 3 blocks and two GU 2 blocks, as illustrated in Fig. 5.

Previously we have seen Yau and Tang's method for realizing universal logic blocks. We will now look at their approach to the design of larger universal networks using smaller universal blocks as the logical connective.

First, any logical function of n variables can be expanded as follows:

$$\begin{aligned} f(x_1, \dots, x_n) &= \bar{x}_1 \bar{x}_2 f(0, 0, x_3, \dots, x_n) + \bar{x}_1 x_2 f(0, 1, x_3, \dots, x_n) \\ &+ x_1 \bar{x}_2 f(1, 0, x_3, \dots, x_n) + x_1 x_2 f(1, 1, x_3, \dots, x_n). \end{aligned}$$

Fig. 5 - Realization of f_2 [g, h, t]

Referring back to Fig. 1, it can be seen that if we let $C_1 \rightarrow X_1$, $C_2 \rightarrow X_2$, and A_0, A_1, A_2, A_3 be tied to $f(0, 0, x_3, \dots, x_n), \dots, f(1, 1, x_3, \dots, x_n)$ respectively, then $f(x_1, \dots, x_n)$ can be realized by a GU 3 block. Now expand the residue function as follows:

$$f(0, 0, x_3, \dots, x_n) = \bar{x}_3 \bar{x}_4 f(0, 0, 0, 0, x_5, \dots, x_n) + \bar{x}_3 x_4 f(0, 0, 0, 1, x_5, \dots, x_n) \\ + x_3 x_4 f(0, 0, 1, 0, x_5, \dots, x_n) + x_3 x_4 f(0, 0, 1, 1, x_5, \dots, x_n).$$

This too can be realized by a GU 3 block with side terminals tied to x_3 and x_4 and front terminals A_0, \dots, A_3 tied to $f(0, 0, 0, 0, x_5, \dots, x_n) \dots f(0, 0, 1, 1, x_5, \dots, x_n)$ respectively.

Similarly, $f(0, 1, x_3, \dots, x_n), \dots, f(1, 1, x_3, \dots, x_n)$ can be expanded and realized by ULM-3 blocks. We repeat this process until the residue functions become functions of x_n alone. We then have essentially realized $f(x_1, \dots, x_n)$ by a tree structure consisting of $(n-1)/2$ levels (assuming n is odd), where there are 4^{j-1} GU 3's in the j^{th} level of the structure. Each of the front terminals of the GU 3 block in the last level is connected to one of the four values $0, 1, x_n, \bar{x}_n$ defined by the corresponding residue function of variable x_n . In this construction there are then $2^{n-1} + n - 1$ total input lines for the resulting universal circuit of n variable. For a more detailed description of this procedure, and for some interesting uses of coding and fault detection tests, the reader is referred to Yau and Tang (9).

Schneider (3), in a well-written and lucid thesis, has considered the problem of synthesizing multiple output logical functions using functional logic blocks as connectives. The resulting synthesis procedure is in the form of a computer algorithm and is based on the theory of cubical complexes and decomposition theory.

It is pointed out that a major problem remaining is that of deciding what function is to be placed on the integrated circuit so as to make the design as efficient as possible. Specifically, the possibility of using universal logic modules is not discussed. However, it is felt that some of Schneider's theory will aid in dealing with this problem. Deciding what subset of universal modules to use will also be of major concern.

PROPOSED INVESTIGATION

As has been seen, most of the work being done today attacks the problem of actually finding logic modules capable of realizing n -variable Boolean functions. A problem still remaining is that of finding the best GU block (i.e., the one with the fewest inputs). Since the lower bound on the number of inputs required for an n -variable universal block increases very rapidly, it appears that the practicality of a universal block in more than four or five variables is limited.

It should be noted that 416 five-variable functions were found that are universal in three variables. It is feasible that by computer search methods seven-variable functions which are universal in four variables can be found. A similar approach used for finding universal functions of five variables seems unfeasible, since there are more than 37×10^6 five-variable equivalence classes. Hence methods must be found which do not require completely exhaustive techniques for finding these functions.

Assuming the availability of a class of universal blocks (let us say in three, four, or five variables), the major problem then is to find optimum synthesis techniques using these blocks. Stated more precisely, we have the following problem: given a Boolean function of n -variables, realize this function using as few blocks as possible.

A number of questions arise when considering the optimum design problem. It would seem that the more uses that can be found for a module, the cheaper the per-unit cost of the module. Now suppose we have three types of universal modules $\{M_A, M_B, M_C\}$, where M_A is universal in two variables, M_B in three variables, and M_C in four variables. That is, each module appears as shown in Fig. 6.

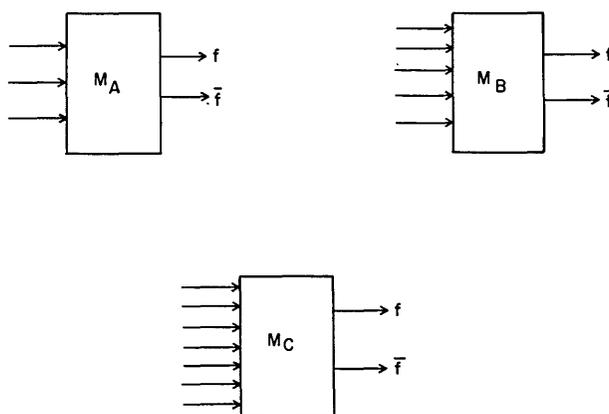


Fig. 6 - Inventory of modules

Now suppose further that there exist K universal functions of two variables, L universal functions of three variables, and m universal functions of four variables (obviously $k < L < m$). Therefore we can consider that the set of possible modules available for design is $\{M_{A1}, M_{A2}, \dots, M_{AK}, M_{B1}, \dots, M_{BL}, M_{L1}, \dots, M_{CM}\}$. It is also assumed that we have a sufficient number of each type available. It is desired to realize a Boolean function of n variables, $f(x_1, \dots, x_n)$, using modules from our available set as logical connectives. (It should be noted that this is indeed possible, since obviously each universal block is also complete.) The overall problem of design can then be broken down by considering the following statements, based upon appropriate constraints.

Constraint 1

All modules will be of a single type (i.e., either M_{A_i} , M_{B_j} or M_{C_k}). If this constraint is observed, the following comments are pertinent.

- What is the minimum number of modules necessary to realize an n variable Boolean function? Does this depend on the particular function in the block?
- Find design techniques to realize the given function using this minimum number of modules (or approach this number as close as possible).
- Since each universal module can be realized using one of many possible universal logic functions, it is conjectured that some functions would be more useful when universal blocks are used in the synthesis of large networks. Also, by mixing different versions of a particular block, ease in design or fewer total modules in the design might result.
- A figure of merit associated with each function which realizes a particular block would be useful, and would aid in the general optimization problem.
- Does the number of logic levels associated with a realization depend upon the functional form of the block used?
- How does a prespecified number of logic levels affect the synthesis procedure?

Constraint 2

Mixing modules of types M_A , M_B , and M_C are to be used in the design. If this constraint is observed, we may ask the following questions.

- It seems that using modules all of type M_C would result in a network of fewer total modules. Is this in fact true, or does there exist some optimum mix?
- How are the number of logic levels affected by mixing modules in the design?
- If a network were to be synthesized in the form of a tree, then it seems that using type M_C modules in the higher levels and type M_A modules in the lower levels would be convenient. Is this indeed true?

With respect to the first question, Elspas, et al. (13), has shown that if only single-output, m -input modules of the same type are used as the logical connective to realize a function of $n > m$ variable, then for $n \gg m$, the choice of a module type is immaterial, when the cost criterion is the number of modules used.

Some General Questions

1. Can we easily synthesize larger universal networks from smaller ones?
2. Some basic building blocks are desirable to have even though they are not universal (i.e., various types of gated flip flops, counters, etc.). Does it pay to keep these on the "shelf" in lieu of synthesizing them from ULB's?
3. By using the inherent time delay of the ULB's and using suitable feedback, sequential machines can be realized. How can we use the properties of ULB's to improve the design of sequential machines? For example, how is the state assignment problem affected?

Perret and David (14) consider the synthesis of sequential circuits using basic modules as a connective.

General Remarks

It seems that the conventional Boolean minimization techniques are not applicable to the optimum design problem. To get a better feel for what an optimum network composed of ULB's "looks like," exhaustive methods seem to help. Suppose, to begin with, it is desired to design an optimum network for a given function of five variables, using blocks which are universal in two variables (i.e., 3 input/2 output blocks). One way to approach this problem is to try all possible combinations of blocks as connectives, together with all possible partitions of input variables until the given function is realized. By increasing the number of blocks by one after each try, the first network to realize the function will contain the minimum number of blocks. This method is similar to the one Hellerman (15) used for NOR and NAND logic. For ULB's of two and three variables, this procedure is feasible. What is hoped is that some insight into the minimization picture will be gained. At the least, we would have obtained a catalog of optimum designs for specific functions.

Since the general problem is one of optimizing the number of blocks, operational research methods of linear and integer programming could be employed to aid in arriving at synthesis methods. If weights or figures of merit could be attached to modules according to universal types and/or functional types, then some realistic optimization criteria could be established. If for each design we could compute a number which indicates its degree of optimization, then we could make decisions based upon these computed numbers.

REFERENCES

1. Earle, J., "What do you put on a Chip?" *Electronic Design*, pp 33-52, 7 Dec. 1964
2. Patt, Y.N., "A Complex Logic Module for the Synthesis of Combinational Switching Circuits," AFIPS Conference Proceedings of the 1967 Spring Joint Computer Conference, pp. 699-705, 1967
3. Schneider, P.R., "Synthesis of Multiple Output Combinational Logic Using a Library of Functionally Packaged Modules," Ph.D. Thesis, Univ. of Wisconsin, 1966 available from University Microfilms (66-9, 968), Ann Arbor, Mich.
4. Forslund, D.C., and Waxman, R., "The Universal Logic Block (ULB) and Its Application to Logic Design," Conference Record of 1966 Seventh Annual Symposium on Switching and Automata Theory
5. Dunham, B., and North, J.H., "The Problem of Logically Efficient Building Blocks and Hookups," IBM Research Paper RC-785, Sept. 18, 1962
6. Dunham, B., et al., "The Multipurpose Bias Device, Part II, The Efficiency of Logical Elements," IBM Journal of Research and Development 3:46, Jan. 1959
7. Elspas, B., et al., "Properties of Cellular Arrays for Logic and Storage," Stanford Research Inst., SR 3, Tech. Report No. AFCRL-67-0463, July 1967
8. Susskind, A.K., et al., "Threshold Elements and the Design of Sequential Switching Networks," MIT Electronic Systems Laboratory Tech. Report No. RADC-TR-67-255, July 1967

9. Yau, S.S., and Tang, C.K., "Universal Logic Circuits and Their Modular Realizations," Proc. AFIPS Conference Proceedings of the 1968 Spring Joint Computer Conference, pp 297-305, 1968
10. King, W.F., and Giusti, A., "The Design of a More Complex Building Block for Digital Systems," Cambridge Research Laboratories AFCRL, 67-0516, Phys. Sci. Res. Paper No. 341, Sept. 1967
11. Case, P.W., et al., "Solid Logic Design Automation," IBM Journal of Research and Development, 8 (2):127-140 (Apr. 1964)
12. Curtis, H.A., "A new approach to Design of Switching Circuits," D. Van Nostrand Company, Inc., Princeton, N.J., 1962
13. Elspas, B., Kautz W.H., and Stone, H.S., "Properties of Cellular Arrays for Logic and Storage," Stanford Research Inst., Final Tech. Report No. AFCRL-68-0005, Nov. 1967.
14. Perret, R., and David, R., "Synthesis of Sequential Circuits Using Basic Cell Elements," Proc. of Ninth Joint Automatic Control Conference, U. of Michigan, pp 582-594, June 1968
15. Hellerman, L., "A Catalog of Three-Variable Or-Invert and And-Invert Logical Circuits," IEEE Trans. Electronic Computers EC-12, pp 198-223, June 1963

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Naval Research Laboratory Washington, D.C. 20390		2a. REPORT SECURITY CLASSIFICATION Unclassified	
		2b. GROUP	
3. REPORT TITLE UNIVERSAL LOGIC BLOCKS			
4. DESCRIPTIVE NOTES (Type of report and inclusive dates) An interim report on one phase of the problem; work is continuing.			
5. AUTHOR(S) (First name, middle initial, last name) Barry P. Shay			
6. REPORT DATE September 17, 1969		7a. TOTAL NO. OF PAGES 20	7b. NO. OF REFS 15
8a. CONTRACT OR GRANT NO. NRL Problem B01-04		9a. ORIGINATOR'S REPORT NUMBER(S) NRL Report 6927	
b. PROJECT NO. RR003-02-41-6150		9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	
c.			
d.			
10. DISTRIBUTION STATEMENT This document has been approved for public release and sale; its distribution is unlimited.			
11. SUPPLEMENTARY NOTES		12. SPONSORING MILITARY ACTIVITY Department of the Navy (Office of Naval Research) Washington, D.C. 20360	
13. ABSTRACT Combinatorial networks are usually synthesized from integrated-circuit building blocks. Advances in integrated-circuit processing technology permit the placement of more logic on a single block, but raise two new questions as to what logical functions should be placed on a block and how combinatorial networks can be efficiently synthesized from a family of blocks. The Universal Logic Block (ULB) provides a promising approach towards answering these questions. A ULB produces a particular function of n Boolean variables (and its complement) that has the property of covering any Boolean function of k or fewer variables by suitable input partitioning. Such functions are known as Globally Universal (GU) functions. Currently available methods of finding GU functions depend upon realizing certain equivalence classes of Boolean functions. The enormous growth rate of these classes limit the practicality of such methods. Techniques must be found which either avoid the realization of equivalence classes, or enlarge each class by defining new relations on the set of Boolean functions. The synthesis of large networks using the GU block as the logical connective remains a difficult problem. By placing constraints on the desired network, realistic design criteria can be established. Considerations to be taken into account include size of the block to be used, delays associated with the block, and the mixing of various types of blocks.			

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Globally universal function Universal logic blocks Decomposition Synthesis Canonical expansion Combinatorial logic Sequential machine Integrated circuit Equivalence class.						