COMPUTER REDUCTION OF BOOLEAN EXPRESSIONS

by

DAVID GORDON DUTRA

B. S. E. E., University of the Pacific
Stockton, California, 1962

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1965

Approved by:

Charles A. Halijak

Charles A. Halijak
Major Professor

TABLE OF CONTENTS

INTRODUCTION

This paper reports on the work of R.W. House and T. Rado (4) on the minimization of Boolean switching functions.

House and Rado have developed several algorithms for obtaining irreducible representations for switching functions using a digital computer with a binary format. Implicit in their work is the subtle difference between a minimum function and an irreducible function. It will be seen that there are no general criteria for a minimum function. The minimal property will depend on the application.

Their approach is to develop a criterion of irreducibility and then to generate all irreducible representations of the given function. The user may then choose the representation that is minimum for his particular application.

There are two types of representations for logical switching functions. One may work with the sum-of-products form (also known as disjunctive form, normal form, or alternational form) or the product-of-sums form (conjunctive form). One is the dual of the other. Ghazala (3) has shown that a simple transformation will convert one to the other. Any algorithm which manipulates the sum-of-products form will, with the help of the Ghazala transformation, manipulate the products-of-sums form.

It was therefore decided to concentrate on one of the two possible types of representations. The sum-of-products form is used in the House-Rado algorithms.

Industry, with the help of Bartee (2), has accepted three criteria to determine the respective minimality between equivalent sum-of-products representations. First, the minimal expression is that expression which contains the least occurrences of literals; second, the minimal expression is the expression containing the least number of product terms; and third, the minimal expression is the expression which requires the least number of diodes in an AND-to-OR circuit configuration. With these in mind, House and Rado have assigned a minimality measure to the irreducible representations, that of a literal count, a term count, and a diode count.

The purpose of this report is to develop in a heuristic manner the House-Rado algorithms. In almost every instance the House-Rado papers have

contained theorem statements, formal proofs, followed by detailed procedures. These writings are mathematical rigorous and follow a trend of expert writing to expert. However, rigor can hamper communication of a subject! Without a lengthy training period a reader is unable to glean the full significance of this outstanding work.

The philosophy of this report is to sacrifice mathematical rigor for orderly development in hope of providing insight to the underlying concepts. To attain this end the next section puts the reader on firm ground as to terminology used; the second section considers the case of multiple input-single output systems; in the third section the case of multiple input-multiple output systems are discussed. In both the second and third sections the subject of completely specified functions is developed and then extended to the "don't care" or incompletely specified functions.

TERMINOLOGY

The terminology and definitions that follow are a meld of those used by Bartee (1), Ghazala (3), Quine (7), and Whitesitt (9). Only Boolean functions of two-valued variables will be considered.

COMMON TERMS

Consider N independent variables

$$x_{N-1}, \cdots, x_n, \cdots, x_0$$

which may take on values 0 or 1.

The complement of a variable is written $\bar{x}_n$ and called a "barred variable".

A literal is a barred or unbarred variable.

The symbol + represents the logical sum (disjunction, inclusive union, inclusive OR).

The symbol • represents the logical product (conjunction, intersection, AND). The • is implied when no symbol is used between literals.

A Boolean function is denoted by f_i, where i takes on different integer values to symbolize different functions.

A term refers to a product of literals; $x_3 \bar{x}_1 x_0$ is a term.

A canonical term is a term containing all N variables (truth table product).

A function expressed in normal form is one expressed in sum-of-products form.

A normal canonical form is a function expressed in sum-of-products form where each product is a canonical term. For N = 3, $f_0 = \bar{x}_2 x_1 \bar{x}_0 + x_2 x_1 x_0$ is a normal canonical form. For every function there exists a single normal canonical form. As an example, consider the function f_1 defined by the truth table of Fig. 1.

Input			Output
x_2	x_1	x_0	f_1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Fig. 1. Truth table defining a Boolean function with a normal canonical form $f_1 = \bar{x}_2 \bar{x}_1 x_0 + x_2 \bar{x}_1 \bar{x}_0 + x_2 x_1 x_0$.

One literal is said to oppose another literal if both literals are of the same variable and one is barred and the other is unbarred; x_n opposes $\bar{x}_n$, and $\bar{x}_n$ opposes x_n.

Two terms oppose if the literals of those terms show opposition; $x_3 \bar{x}_1 x_0$ and $x_3 \bar{x}_2 x_1$ show one opposition, $x_3 \bar{x}_0$ and $x_5 \bar{x}_3 x_2 x_0$ show two oppositions.

The consensus of two terms exist if the terms show exactly one opposition. The consensus is found by forming the product of the two terms and deleting the two literals of the single opposition; the consensus of $\bar{x}_3 x_2 x_1$ and $x_3 x_1 x_0$ is $x_2 x_1 x_0$.

A term t_1 is said to subsume a term t_2 if all the literals of t_2 are contained in t_1; $\bar{x}_3\, x_1\, \bar{x}_0$ subsumes $x_1\, \bar{x}_0$ and $\bar{x}_5\, x_0$ subsumes x_0.

IMPLICATION

There is one concept that is used extensively through out this report that is worth amplifying. That is the concept of a partial equality termed implication. To put this on familiar ground consider the switching circuit of Fig. 2.

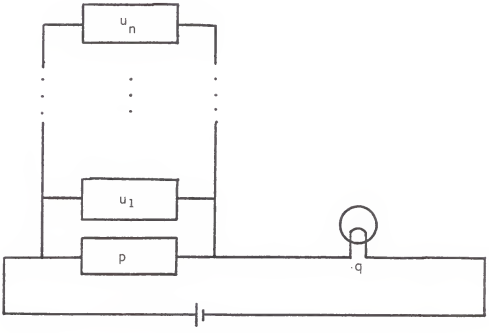

Fig. 2. A switching circuit to demonstrate the concept of implication.

In Fig. 2 let u_i, $i = 1, \ldots, n$, and p be generic variables (represents any switching function or variable) and $q = 1$ when the lamp is lit and $q = 0$ when the lamp is dark. A complete description of q would be

$$q = p + \sum_{i=1}^{n} u_i$$

Suppose that the nature of all u_i, i = 1, ..., n, is unknown and it is desired to express a relationship between p and q. It could be said that if p = 1, then q = 1, or simply that if p then q. This is known to logicians as material implication or simply implication, i.e., p implies q. p being equal to 1 implies that q is equal to 1. Nothing is said about the converse in this statement of implication! If q = 1, nothing is known about p, it may equal 1 or 0.

Fig. 3 lists all possible values of p and q such that p implies q (symbolized: $p \rightarrow q$).

Row	p	q
1.	0	0
2.	0	1
~~3.~~	~~1~~	~~0~~
4.	1	1

Fig. 3. Permissible values of p and q for $p \rightarrow q$.

Note that for $p \rightarrow q$ Row 3 cannot appear in the table of all possible combinations of p and q; p and q cannot simultaneously equal 1 or 0, respectively. It may be said that $p \rightarrow q$ means that if p = 1 then q = 1 or $\bar{p} + q = 1$, $pq = p$, $q + p = q$, etc.

It follows then, that for any normal form of a function f each term of f implies f, i.e., for

$$f = f_1 + \cdots f_M,$$

$$f_1 \rightarrow f$$

$$\vdots$$

$$f_M \rightarrow f$$

Equality may be considered a double implication; that is, if $f_1 = f_2$, then $f_1 \rightarrow f_2$ and $f_2 \rightarrow f_1$.

MULTIPLE INPUT-SINGLE OUTPUT SYSTEMS

A multiple input-single output system is a logical switching network with N inputs and a single output represented by the Boolean function

$$f = f(x_{N-1}, \ldots, x_0)$$

As noted in the Introduction, f will be assumed to be in a normal form.

COMPLETELY SPECIFIED FUNCTIONS

A completely specified function is one in which an output value, 1 or 0, is defined for every possible value of the N input variables x_{N-1}, ..., x_0.

Prime Implicants

If f is formed from its truth table representation without simplification, it will be in normal canonical form; f will be expressed as a sum of its canonical implicants. By definition, each term will contain all N variables, the maximum size of a term for a given system, and there will be one term for each 1-valued output of the system. This is a maximum normal form of f. Each term is of maximum size and all possible disjoint terms appear. It may be considered the maximum form for a sum of disjoint terms. Canonical terms are disjoint in the sense that each one implies only a single 1-valued output.

There are a finite number of unique implicants for any given system. The maximum possible formed from N independent variables is $3^N - 1$ implicants. Suppose that one of the non-canonical implicants, say t_i, is added to the canonical form of f. By definition, t_i will contain fewer literals than any canonical implicant. Also, t_i will be implied by at least two of the canonical terms. This is so because t_i has at most N-1 literals and must imply at least two 1-valued outputs of the system.

Therefore the presence of t_i will cause the canonical terms that imply t_i to be redundant and they may be deleted. The resulting expression will contain at least one fewer terms and a term with at least one fewer literals.

Suppose, instead of adding a single non-canonical implicant to the canonical form of f, that all non-canonical implicants were added and that all terms that implied another were then deleted. The resulting representation would be a sum, to represent the system, of terms containing the fewest possible literals. These terms are called the prime implicants of f.

As an example, consider a system where N = 2 and f is defined by Fig. 4.

All possible implicants for N = 2 are listed in Fig. 5.

Input		Output
x_1	x_0	f
0	0	1
0	1	1
1	0	1
1	1	0

Fig. 4. Truth table representation of a function f.

Input		Possible implicants for a system of N = 2							
x_1	x_0	x_1	x_0	$\bar{x}_1$	$\bar{x}_0$	x_1x_0	$x_1\bar{x}_0$	$\bar{x}_1x_0$	$\bar{x}_1\bar{x}_0$
0	0	0	0	1	1	0	0	0	1
0	1	0	1	1	0	0	0	1	0
1	0	1	0	0	1	0	1	0	0
1	1	1	1	0	0	1	0	0	0
Column		1	2	3	4	5	6	7	8

Fig. 5. All possible implicants for a system of two variables.

By comparing the Columns of Fig. 5 to f of Fig. 4, it is seen that the terms of Columns 1, 2, and 5 of Fig. 5 must be cast out as not implying f. What remains are implicants of f and are tabulated in Fig. 6.

Input		Implicants				
x_1	x_0	$\bar{x}_1$	$\bar{x}_0$	$x_1\bar{x}_0$	$\bar{x}_1x_0$	$\bar{x}_1\bar{x}_0$
1	0	1	1	0	0	1
1	1	1	0	0	1	0
0	0	0	1	1	0	0
0	1	0	0	0	0	0
Column		1	2	3	4	5

Fig. 6. Implicants of the function defined in Fig. 4.

In Fig. 6 the implicants of Columns 3, 4, and 5 imply those of 1 and 2 and may therefore be deleted. The prime implicants of f are then those implicants of Columns 1 and 2, $\bar{x}_1$ and $\bar{x}_0$. These are the terms that imply f, do not imply each other, and are sufficient to represent f as a sum.

$$f = \bar{x}_1 + \bar{x}_0 \qquad (1)$$

Equation (1) is the single irreducible representation of f. In general this does not always happen. For N > 2 there will usually be more than one irreducible representation of the system, i.e., some prime implicants may imply sums of other prime implicants.

Consider the case of an f_1 expressed as the sum of its prime implicants.

$$f_1 = x_2\bar{x}_1 + x_1\bar{x}_0 + \bar{x}_2x_0 + \bar{x}_2x_1 + \bar{x}_1x_0 + x_2\bar{x}_0 \tag{2}$$

Notice in equation (2) that

$$x_2\bar{x}_1 \rightarrow \bar{x}_1x_0 + x_2\bar{x}_0 \tag{3}$$

$$x_1\bar{x}_0 \rightarrow \bar{x}_2x_1 + x_2\bar{x}_0 \tag{4}$$

$$\bar{x}_2x_0 \rightarrow \bar{x}_2x_1 + \bar{x}_1x_0 \tag{5}$$

$$\bar{x}_2x_1 \rightarrow x_1\bar{x}_0 + \bar{x}_2x_0 \tag{6}$$

$$\bar{x}_1x_0 \rightarrow \bar{x}_2x_0 + x_2\bar{x}_1 \tag{7}$$

$$x_2\bar{x}_0 \rightarrow x_2\bar{x}_1 + x_1\bar{x}_0 \tag{8}$$

Any normal form of f_1 containing $\bar{x}_1x_0 + x_2\bar{x}_0$ causes $x_2\bar{x}_1$ to be redundant in that representation, equation (3); any representation containing $\bar{x}_2x_1 + x_2\bar{x}_0$ will be redundant if $x_1\bar{x}_0$ is also present, equation (4). Similar statements may be made using equations (5), (6), (7), and (8).

It is shown by House-Rado (5) that equation (2) has five equivalent representations which contain fewer terms. They are:

$$\begin{aligned}
f_1 &= x_2\bar{x}_0 + \bar{x}_1x_0 + \bar{x}_2x_1 \\
f_1 &= \bar{x}_2x_0 + x_1\bar{x}_0 + x_2\bar{x}_1 \\
f_1 &= x_2\bar{x}_0 + \bar{x}_1x_0 + \bar{x}_2x_0 + x_1\bar{x}_0 \\
f_1 &= x_2\bar{x}_0 + \bar{x}_2x_1 + \bar{x}_2x_0 + x_2\bar{x}_1 \\
f_1 &= \bar{x}_1x_0 + \bar{x}_2x_1 + x_1\bar{x}_0 + x_2\bar{x}_1
\end{aligned} \tag{9}$$

Note that the expressions of equation (9) are still sums of the prime implicants of f_1, but that in each case one or more prime implicants have been deleted. These representations are the irreducible (irredundant) forms of f_1, irreducible in the sense that if any term were deleted the resulting function would cease to express f_1.

It may now be stated that the minimization philosophy for single output systems is first, find all the prime implicants of the system function, which is in reality finding the terms that will express the system function and have the fewest possible literals. And second, find all the irreducible representations of the system as sums of prime implicants. Stated in other words, to minimize a single Boolean function, one first minimizes the

literal count per term and then the term count.

Calculation of Prime Implicants

To proceed in the manner outlined would be a very lengthy process even for a high speed digital computer. The value of the expression $3^N - 1$ increases rapidly with N. For example when N = 8, it would require investigation of 6,560 implicants just to generate the prime implicants. Fortunately Quine (7) has discovered an algorithm that will generate all the prime implicants of a Boolean expression in an almost automaton manner.

The Quine algorithm may be developed in the following manner. Consider a function

$$f = f_1 + \cdots + f_m + \cdots + f_M \tag{10}$$

where each f_m, $1 \leq m \leq M$, is a Boolean term. It is desired to find all the prime implicants of equation (10).

Any f_m that implies another f_m, $1 \leq m \leq M$, is, by definition, not a prime implicant of f. So as a first step, all terms may be cast out that imply other terms. These are easily found because the only way one Boolean term can imply another is for the former to subsume the latter.

Now, any term that remains is either a prime implicant of f or it subsumes some prime implicant that is not present. If there is a prime implicant absent, say p, there must be terms x_n p and $\bar{x}_n$ p among the terms that remain, because $p = x_n p + \bar{x}_n p$. If these two were not present, p could not imply f, but it must by definition. Therefore any prime implicant of f that is not present in the original expression will appear as the consensus of two terms that are present in the original expression.

The Quine algorithm may be stated as follows.

A Boolean function in normal form goes into a sum of its prime implicants by alternating the following two equivalence transformations until they are no longer applicable.

i) If a term subsumes another delete the former.

ii) Add to the resulting terms the consensus of two terms unless it subsumes a term already present.

As an example consider

$$f = x_2\bar{x}_1 + x_1\bar{x}_0 + x_2\bar{x}_1x_0 + \bar{x}_2x_1x_0 + \bar{x}_2x_0 \tag{11}$$

Apply i) to equation (11) and obtain

$$f = x_2\bar{x}_1 + x_1\bar{x}_0 + \bar{x}_2x_0 \tag{12}$$

Apply ii) to equation (12) and obtain

$$f = x_2\bar{x}_1 + x_1\bar{x}_0 + \bar{x}_2x_0 + x_2\bar{x}_0 + \bar{x}_1x_0 + \bar{x}_2x_1 \tag{13}$$

Apply i) and ii) to equation (13) and find that there is no change and that f is expressed as a sum-of-prime implicants.

It is obvious that the Quine algorithm is easily implemented on a digital computer.

Logically the next step is to develop a procedure for finding all irreducible representations of an expression. The procedure to be developed will not be restricted to functions expressed as sums of prime implicants. In fact, it is not required that the function be a sum-of-terms. The procedure is valid for any expression that is a sum-of-functions, Rado (8).

Irreducible Representations

Consider a Boolean expression of the form

$$f = f_0 + \cdots + f_m + \cdots + f_{M-1} \tag{14}$$

where each f_m, $0 \leq m \leq M-1$, is a function of the N variables previously noted.

It is desired to find all the irreducible forms of equation (14). A representation f is called irreducible if no proper subset of $f_0, \ldots, f_{M-1}$ as a sum is a representation for f. (A proper subset is a subset that is not all inclusive.)

For simplicity begin with M = 4 and then extend this to the general case. When M = 4,

$$f = f_0 + f_1 + f_2 + f_3 \tag{15}$$

Fig. 7 lists all the unrestricted combinations of f_0, f_1, f_2, and f_3 for equation (15).

Row	f_0	f_1	f_2	f_3
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Fig. 7. All unrestricted combinations of function-values for equation (15).

By unrestricted, it is meant that this is the case where all possible combinations are present. Obviously when this is the case, f is already irreducible; there is no proper subset of $f_0, \ldots, f_{M-1}$ that will represent f. Therefore if f is reducible, there must be one or more rows of Fig. 7 that are missing, i.e., some one or more of f_m must imply some other f_m or a sum of them, $0 \leq m \leq M-1$. Suppose that in equation (15) $f_0 \rightarrow f_1 + f_2$ and $f_2 \rightarrow f_0 + f_3$. This restriction would eliminate Rows* 2, 6, 8, and 9 as is shown in Fig. 8.

Row	f_0	f_1	f_2	f_3
0	0	0	0	0
1	0	0	0	1
*2				
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
*6				
7	0	1	1	1
*8				
*9				
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Fig. 8. All possible combinations of function-values for equation (15) when restricted by the implications $f_0 \rightarrow f_1 + f_2$ and $f_2 \rightarrow f_0 + f_3$.

Now, define a binary presence variable p_m associated with each f_m, where $p_m = 1$ when f_m is present in any representation of f and $p_m = 0$ when f_m is not present in that representation, $0 \leq m \leq M-1$.

Each included row except Row 0 of Fig. 5 indicates a condition for f = 1. Only a single 1 in a row is necessary to specify that f = 1. Therefore each row indicates the sufficient-presence of $f_0, \ldots, f_{M-1}$ to cause f = 1 in any representation of f. From Row 1: the presence of f_3 is required, Row 3: the presence of f_2 or f_3 is required, Row 4: the presence of f_1 is required, etc. The list of sufficient conditions for any representation of equation (15) may be expressed in Boolean form as

$$
\begin{aligned}
&p_3 = 1\\
&p_2 + p_3 = 1\\
&p_1 = 1\\
&p_1 + p_3 = 1\\
&p_1 + p_2 + p_3 = 1\\
&p_0 + p_2 = 1\\
&p_0 + p_2 + p_3 = 1\\
&p_0 + p_1 = 1\\
&p_0 + p_1 + p_3 = 1\\
&p_0 + p_1 + p_2 = 1
\end{aligned}
\qquad (16)
$$

Note in equation (16) that no presence-sum is required for Row (15) of Fig. 8. It is necessary that at least one of the f_m, $0 \leq m \leq M-1$, be present for f not to equal 0 for all values of the independent variables, a trivial case that is ignored.

The necessary condition for any representation of f is that all the sufficient conditions be simultaneously satisfied. If the Boolean product of the left sides of equation (16) is taken, it will equal 1 for each representation of f. Let this product be called the presence function S.

$$
\begin{aligned}
S = p_3\,(p_2 + p_3)\,p_1\,(p_1 + p_3)(p_1 + p_2 + p_3)(p_0 + p_2)(p_0 + p_2 + p_3)(p_0 + p_1)\\
(p_0 + p_1 + p_0)(p_0 + p_1 + p_2)
\end{aligned}
\qquad (17)
$$

When S is expanded into a normal form, it is obvious that S = 1 when any term is equal to 1. A term is equal to 1 only when each literal has a value of 1. Therefore each term of S will indicate a representation of the given expression f. Each representation will consist of a sum of those f_m, $0 \leq m \leq M-1$, associated with the presence variables p_m, $0 \leq m \leq M-1$, of a term. For instance if a term of S were $p_0p_2p_4$, a representation of the given expression f would be

$$f = f_0 + f_2 + f_4$$

If S is reduced so that each term contains the fewest possible literals, it will indicate representations of f that are sums of the fewest possible f_m, $0 \leq m \leq M-1$, and are then irreducible representations. S takes on

this form when it is expressed as a sum-of-prime implicants.

$$S = p_0p_1p_3 + p_1p_2p_3 \tag{18}$$

is equation (17) expressed as a sum-of-prime implicants and the irreducible representations of f are

$$\begin{aligned} f &= f_0 + f_1 + f_3 \\ f &= f_1 + f_2 + f_3 \end{aligned} \tag{19}$$

Note a fortunate occurrence. S contains no barred variables; therefore when the Quine algorithm is applied, consensus products are not possible and it is necessary only to delete subsuming terms to generate the prime implicants.

The next step is to develop a computer adaptable scheme for generating the House-Rado presence function for the general case of M functions.

Consider what the presence function S, equation (17), really is. It is a product-of-sums, where each sum is obtained from a non-zero row of Fig. 8. Figure 8 is a table of values taken on by f_m of equation (14), $0 \leq m \leq M-1$, as the set of independent variables, $x_{N-1}, \ldots, x_0$, range over all possible values.

Let u_i represent the binary equivalent of the decimal integer i, and let $f_m(u_i)$ represent the substitution of u_i for the independent variables $x_{N-1} \cdots x_0$ in the function f_m.

All the presence-sums of equation (16) will be generated by allowing i to vary from 0 to $2^N - 1$ in

$$\begin{aligned} A_i &= f_0(u_i)p_0 + \cdots + f_{M-1}(u_i)p_{M-1} \\ &= \sum_{m=0}^{M-1} f_m(u_i)p_m, \quad i = 0, 1, \ldots, 2^N - 1 \end{aligned} \tag{20}$$

Each i will generate a possibly non-unique presence-sum. This is to be expected because several values of the independent variable could map into the same row of Fig. 8. Also, at least one, and probably several, values of i will cause $A_i = 0$ corresponding to Row 0 of Fig. 8. If it were not for these zeros the presence function S could be formed directly as a product of all A_i, $0 \leq i \leq 2^N - 1$.

Define a generalized Kronecher Delta as

$$\Delta\ (V_0, \ldots, V_{M-1}) = \begin{cases} 1 \text{ if } V_{M-1} = \cdots = V_0, \\ 0 \text{ otherwise} \end{cases} \tag{21}$$

where $V_{M-1}, \ldots, V_0$ are Boolean variables.

Non-zero presence-sums then become

$$B_i = \Delta\ [f_0(u_i), \ldots, f_{M-1}(u_i)] + \sum_{m=0}^{M-1} f_m(u_i)p_m,$$
$$i = 0, 1, \ldots, 2^N - 1 \tag{22}$$

and finally the presence function

$$S(p_0, \ldots, p_{M-1}) = \prod_{i=0}^{2^N-1} B_i \tag{23}$$

Notice that the Δ function in equation (22) excludes the two cases of no interest, those corresponding to Rows 0 and 15 of Fig. 8.

INCOMPLETELY SPECIFIED FUNCTIONS

Oftentimes the independent variables of a switching system are by the nature of the system restricted from taking on all possible values. There can be input combinations that never occur. Systems with this trait are represented by switching functions referred to as incompletely specified functions, House-Rado (6), or representations modulo don't cares, House-Rado (4).

A representation modulo don't cares is a function

$$f = f(x_{N-1}, \ldots, x_0), \text{ mod } D \tag{24}$$

where mod D signifies that equation (24) is defined for all substitutions $u \notin D$. D is the set of truth table substitutions for which f is not specified and u is the generic notation for an N digit binary number representing an independent variable substitution.

Consider as was done in equation (14) a representation of a sum-of-functions, but now let it be a representation mod D.

$$f = f_0 + \cdots + f_m + \cdots + f_{M-1}, \text{ mod } D \tag{25}$$

where each f_m, $0 \leq m \leq M-1$, is a function of the N independent variables previously noted and D is the set of don't cares. It is desired to form all irreducible representations mod D of f.

By definition a representation mod D is one that is defined for all inputs that are not a member of the don't care inputs, the set D. Therefore an irreducible mod D representation of f, equation (25), is a mod D representation which ceases to express f mod D when any f_m, $0 \leq m \leq M-1$, is dropped. Then the only difference between equations (14) and (25) is that in equation (14) the independent variables are allowed to range over all 2^N possibilities and in equation (25) they are restricted to values not in the set D. A table comparable to Fig. 8 for equation (25) would then be obtained if u were allowed to vary over all $u \notin D$ in equation (25).

Remembering that Fig. 8 was the basic element used in forming the presence function equation (23), then exactly the same reasoning can be used to formulate a presence function for mod D representations. The only modification required in the development is that i in equations(20), (22), and (23) be restricted to values such that $u_i \notin D$. Then equation (23) becomes

$$S(p_0, \ldots, p_{M-1}) = \prod_{\text{all } u \notin D} \{\Delta[f_0(u), \ldots, f_{M-1}(u)] + \sum_{m=0}^{M-1} f_m(u) p_m\} \quad (26)$$

Equation (26) is the presence function for mod D representations.

Consider the consequence of allowing the set D in equation (26) to be null. Equations (23) and (26) are then identical. A null set D indicates that f is completely specified. Therefore equation (26) will indicate all irreducible representation of any function, if a completely specified function is assumed to have a null set of don't cares.

If equation (26) is to be applied to a function represented by a truth table, one further simplification occurs. The Δ function in the product of equation (25) causes a 1 to appear for any u such that f(u) = 0 and if these values of the independent variable were neglected, nothing would be lost. So equation (26) may also be written as

$$S(p_0, \ldots, p_{M-1}) = \prod_{\text{all } u \in f = 1} \{\Delta[f_0(u), \ldots, f_{M-1}(u)] + \sum_{m=0}^{M-1} f_m(u) p_m\} \quad (27)$$

where all $u \in f = 1$ means that the product is taken over all values of the input that causes an output of value 1.

When a mod D system is being designed, often a great savings in hardware may be accomplished by assuming that some of the inputs contained in the set D yield an output of value 1. This is easily seen by remembering

that the first step in constructing minimum representations of a system is to express the system function as a sum-of-terms containing the fewest literals, these being the prime implicants. The greater the number of terms in an original representation, the greater is the possibility that the Quine algorithm will generate small terms. The chance of creating multiple consensus products is increased. Therefore if the set D is assumed to yield 1-valued outputs there is a chance that the resulting set of prime implicants will contain terms of fewer literals than would otherwise have occurred. Of course, this procedure could create superflous prime implicants and does in most cases. These, however, will be deleted as redundant terms when the presence function is applied because the presence function indicates representations with the minimum number of terms.

With the aid of the Quine algorithm, the House-Rado presence function (equation [26] or [27]) will generate all minimal-literal, minimal-term forms of a single-output switching system on a binary digital computer. Also, the presence function alone may be used to generate all irreducible representations of a sum of Boolean functions. This can be used to advantage by persons testing large scale check-out systems, such as go-no-go missile launch systems as noted by House-Rado (5) and Battelle (2).

REVIEW AND EXAMPLE

The general procedure for finding all irreducible, minimal-literal term, representations of a single-output switching system may be stated as follows.

1) Generate the system prime implicants from the given representation, either functional or truth table, by using the Quine algorithm with the assumption that all inputs of set D yield an output of value 1. Set D will be null for a completely specified function.

2) Find the prime implicants of the presence function, equation (26) or (27), using the sum of the prime implicants found in 1) as f.

3) Create an irreducible representation from every prime implicant found in 2).

The following example taken from House-Rado (6) demonstrates this procedure.

Figure 9 lists the specified outputs for all possible input combinations and Fig. 10 lists the impossible inputs. That is, Fig. 10 defines the set D. All irreducible mod D representations of the output f are to be found.

Input				Output
x_3	x_2	x_1	x_0	f
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	1	0	0	0
0	1	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Fig. 9. Specified Outputs.

Input			
x_3	x_2	x_1	x_0
0	0	1	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	1	0	0

Fig. 10. The set D.

From Fig. 9 and Fig. 10 the set of canonical terms are found to be used to generate the prime implicants and are listed in Fig. 11.

Canonical Terms			
$\bar{x}_3$	$\bar{x}_2$	x_1	$\bar{x}_0$
x_3	$\bar{x}_2$	x_1	$\bar{x}_0$
x_3	$\bar{x}_2$	x_1	x_0
x_3	x_2	$\bar{x}_1$	x_0
x_3	x_2	x_1	$\bar{x}_0$
$\bar{x}_3$	$\bar{x}_2$	x_1	x_0
$\bar{x}_3$	x_2	x_1	$\bar{x}_0$
$\bar{x}_3$	x_2	x_1	x_0
x_3	$\bar{x}_2$	$\bar{x}_1$	$\bar{x}_0$
x_3	$\bar{x}_2$	$\bar{x}_1$	x_0
x_3	x_2	$\bar{x}_1$	$\bar{x}_0$

Fig. 11. Terms to generate the prime implicants.

When the Quine algorithm is applied to the terms of Fig. 11 the prime implicants are found and tabulated in Fig. 12.

f_0: $\bar{x}_3 x_1$

f_1: $x_3 \bar{x}_1$

f_2: $x_3 \bar{x}_2$

f_3: $x_3 \bar{x}_0$

f_4: $\bar{x}_2 x_1$

f_5: $x_1 \bar{x}_0$

Fig. 12. Prime Implicants.

Figure 12 indicates that

$$f = f_0 + f_1 + f_2 + f_3 + f_4 + f_5 \tag{28}$$

as a sum-of-prime implicants. The presence function may now be applied to equation (28). Each sum of equation (27) is tabulated in Fig. 13 using the inputs that yield 1-valued outputs as specified in Fig. 9.

Input u for f(u) = 1				Values of prime implicants $f_m(u)$							$\Delta + \sum_{m=0}^{M-1} f_m(u)p_m$
x_3	x_2	x_1	x_0	f_0	f_1	f_2	f_3	f_4	f_5	Δ	
0	0	1	0	1	0	0	0	1	1	0	$p_0 + p_4 + p_5$
1	0	1	0	0	0	1	1	1	1	0	$p_2 + p_3 + p_4 + p_5$
1	0	1	1	0	0	1	0	1	0	0	$p_2 + p_4$
1	1	0	1	0	1	0	0	0	0	0	p_1
1	1	1	0	0	0	0	1	0	1	0	$p_3 + p_5$

Fig. 13. Sums in the presence function equation (27).

The product of the rightmost column of Fig. 13 is

$$S = (p_0 + p_4 + p_5)(p_2 + p_3 + p_4 + p_5)(p_2 + p_4)\, p_1\, (p_3 + p_5) \tag{29}$$

Equation (29) expressed as a sum-of-prime implicants is

$$S = p_0p_1p_2p_3 + p_1p_4p_5 + p_1p_3p_4 + p_1p_2p_5 \tag{30}$$

Equation (30) indicates the four irreducible mod D representation of f as

$p_0p_1p_2p_3$: $f = f_0 + f_1 + f_2 + f_3$

$p_1p_4p_5$: $f = f_1 + f_4 + f_5$

$p_1p_3p_4$: $f = f_1 + f_3 + f_4$

$p_1p_2p_5$: $f = f_1 + f_2 + f_5$

Refer to Fig. 12 and find that:

$$f = \bar{x}_3x_1 + x_3\bar{x}_1 + x_3\bar{x}_2 + x_3\bar{x}_0$$
$$f = x_3\bar{x}_1 + \bar{x}_2x_1 + x_1\bar{x}_0$$
$$f = x_3\bar{x}_1 + x_3\bar{x}_0 + \bar{x}_2x_1$$
$$f = x_3\bar{x}_1 + x_3\bar{x}_2 + x_1\bar{x}_0$$

MULTIPLE INPUT-MULTIPLE OUTPUT SYSTEMS

This section considers binary switching systems with N inputs and M outputs according to House-Rado (1). A multiple input-multiple output switching system will be represented by a system of functions

$$S: \quad f_{M-1}, \ldots, f_m, \ldots, f_0 \tag{31}$$

where each f_m, $0 \leq m \leq M-1$, is a function of the N independent variables $x_{N-1}, \ldots, x_n, \ldots, x_0$. Let there be M binary variables y_m, $0 \leq m \leq M-1$, such that $y_m = f_m$ for all m, $0 \leq m \leq M-1$, that is, y_m is the value of the mth output of the switching system.

COMPLETELY SPECIFIED FUNCTIONS

As in the last section, begin with the completely specified system; that is, for each of 2^N possible input combinations every f_m, $0 \leq m \leq M-1$, of equation (31) has a defined value, 0 or 1.

At first glance it might seem reasonable to construct minimal representations of the system S by considering each f_m of equation (31) separately and finding all irreducible representations of each. However, upon investigation of this procedure it will be found that it would neglect any attempt at maximizing the number of implicants common to two or more f_m, $0 \leq m \leq M-1$. Any implicant that is common to two or more of the functions of the system S should be retained because a single piece of hardware could be used to implement that term no matter how many times it appeared. Therefore, the procedure to be developed will find minimum-literal implicants of the system S as a whole, saving those implicants that imply more than one f_m, $0 \leq m \leq M-1$. It will be seen that the end result of this procedure will formulate at least three minimum representations of the system S, those representions having 1) a minimum number of unique terms, 2) a minimum number

of literals contained in all unique terms, and 3) the minimum number of diodes required to construct the system in an AND-to-OR circuit configuration.

Prime Origins

Begin by considering a seemingly trivial set of tautologies

$$\begin{aligned} f_{M-1} + \bar{y}_{M-1} &= 1 \\ &\vdots \\ f_m + \bar{y}_m &= 1 \\ &\vdots \\ f_0 + \bar{y}_0 &= 1 \end{aligned} \tag{32}$$

Any manipulation done within the framework of equation (32) must necessarily be a manipulation of the system S (equation [31]). The set of equations (32) may be expressed more simply by

$$A^- = \prod_{m=0}^{M-1} (f_m + \bar{y}_m) = 1 \tag{33}$$

which is a House-Rado address-function.

Then it may be said that any representation of A^- for which $A^- = 1$ will indicate a representation of the system S. This logic parallels that of the formulation of the House-Rado presence function (equation [26]).

Upon carrying out the product indicated by equation (33) there will exist three types of terms; i.e., A^- will be implied by three distinctive implicants. They are: 1) terms containing a product of x and $\bar{x}$ literals times a product of $\bar{y}$ literals; 2) terms containing only a product of x and $\bar{x}$ literals; and 3) terms containing only the product of $\bar{y}$ literals. For conciseness let each implicant be symbolized by PQ, where P is the generic notation for a product of x and $\bar{x}$ literals and Q is the generic notation for a product of $\bar{y}$ literals. A null Q or P will be considered to be equal to 1 as in cases 2) or 3) respectively.

Any term of A^-, PQ, that implies that $A^- = 1$ will imply the presence of

a term in the system S. Consider an implicant PQ, case 1). P implies that $A^- = 1$ whenever $Q = 1$. $Q = 1$ only if each $\bar{y}_m \in Q$ is equal to 1 or that $y_m = 0$ for all m such that $\bar{y}_m \in Q$. When $y_m = 0$, $f_m = 0$ by definition. P cannot imply a function that is equal to 0, so P must imply all those f_m of the system S where m is such that $\bar{y}_m \notin Q$. Therefore the P portion of an implicant PQ of A^- appears in every f_m of the system S for which m is such that $\bar{y}_m \notin$ that Q.

It follows then for case 2), a null Q, that an implicant P of A^- will appear in every f_m of equation (31). Case 3), that of a null P, can be seen to be trivial. It will appear only once in any A^- and Q will contain all $\bar{y}_m$, $0 \leq m \leq M-1$; this may be interpreted as meaning that the null product of x and $\bar{x}$ literals implies no f_m, $0 \leq m \leq M-1$.

Suppose that equation (33) is expressed as a sum-of-prime implicants. Each prime implicant will indicate a term of the system S that contains the fewest possible literals. Notice that this process will not delete a term common to several f_m, $0 \leq m \leq M-1$, even if it is subsumed by an implicant common to only one of those f_m. This is true because of the presence of the Q portion of the A^- implicants. No implicant of A^- can subsume another unless the P and Q portions of the former subsume both the P and Q portions of the latter.

Therefore, the prime implicants of A^- will contain P portions that imply each f_m, $0 \leq m \leq M-1$, individually and P portions that imply groups of f_m, $0 \leq m \leq M-1$, each with as few literals as possible. These P portions of the prime implicants of A^- are then the set of all terms necessary to write all representations of the system S using terms with the fewest possible literals. This set of terms is called the prime origins of the system S. Prime origins are analogous to prime implicants of single-output systems.

As an example consider a switching system with two outputs and three inputs represented by the system of functions

$$\begin{aligned} S: \quad f_1 &= \bar{x}_2 x_0 + x_1 x_0 + \bar{x}_1 \bar{x}_0 \\ f_0 &= x_2 x_0 + \bar{x}_1 \bar{x}_0 + \bar{x}_2 x_1 \end{aligned} \tag{34}$$

By equation (33)

$$A^- = (\bar{x}_2x_0 + x_1x_0 + \bar{x}_1\bar{x}_0 + \bar{y}_1)(x_2x_0 + \bar{x}_1\bar{x}_0 + \bar{x}_2x_1 + \bar{y}_0)$$

$$= \bar{x}_2x_0\bar{y}_0 + x_1x_0\bar{y}_0 + \bar{x}_1\bar{x}_0\bar{y}_0 + x_2x_0\bar{y}_1 + \bar{x}_1x_0\bar{y}_1$$

$$+ \bar{x}_2x_1\bar{y}_1 + \bar{x}_2x_1x_0 + x_2x_1x_0 + \bar{x}_2x_1x_0 + \bar{x}_1\bar{x}_0$$

Using the Quine algorithm the prime implicants of A^- are found to be $\bar{y}_1\bar{y}_0$, $\bar{x}_2\bar{x}_1\bar{y}_0$, $\bar{x}_2x_0\bar{y}_0$, $\bar{x}_2x_1\bar{y}_1$, $\bar{x}_2\bar{x}_0\bar{y}_1$, $x_2\bar{x}_1\bar{y}_1$, $x_2x_0\bar{y}_1$, $\bar{x}_1\bar{x}_0$, and x_1x_0. The P portions of these terms are, by definition, the prime origins of the system S and are listed in Fig. 14.

Prime Origins	
Implicants of f_0	Implicants of f_1
$\bar{x}_2x_1$	$\bar{x}_2\bar{x}_1$
$\bar{x}_2\bar{x}_0$	$\bar{x}_2x_0$
$x_2\bar{x}_1$	$\bar{x}_1\bar{x}_0$
x_2x_0	x_1x_0
$\bar{x}_1\bar{x}_0$	
x_1x_0	

Fig. 14. Prime origins of equation (34).

Figure 14 lists those terms that are sufficient to represent the system S with the properties that they are minimum-literal terms and that all terms common to both f_0 and f_1 are included. All irreducible representations of the system S are sums of these prime origins.

Calculation of Prime Origins

For a system S with many functions, the formation of the address-function A^- using equation (33) can be a tedious process. When S is given as a truth table representaiton there exists a very simple procedure for forming A^-.

Suppose that A^- is to be represented directly in normal form. Again let each term of A^- be noted generically as PQ. The nature of the function A^- dictates that $A^- = 1$ for every possible independent variable combination; it is a tautology. It is also known that each term of A^- has a particular form. That is when the P portion of a term implies one or more f_m, $0 \leq m \leq M-1$,

the Q portion of that term lacks all $\bar{y}_m$, m such that P implies f_m, and includes all other $\bar{y}_m$, $0 \leq m \leq M-1$.

Therefore A^- may be written in normal form directly from a truth table representation of the system S. A^- is the sum of terms where every term contains a P portion that is a canonical product of x and $\bar{x}$ literals and a Q portion that is a product containing a $\bar{y}_m$ for every f_m that has a value of 0 for that canonical P. The sum must include a term formed from every row of the truth table to insure that A^- is a tautology. The original expression for A^-, equation (33), includes a term, $\bar{y}_{M-1} \cdots \bar{y}_0$, representing the occurrence of a null set of input variables. This term cannot contribute any products to the set of prime origins, but should be included for reasons that will become clear in a future example.

Define

$$Q_P = \prod \bar{y}_m \tag{35}$$

where the product is taken over all m such that P does not imply f_m; or equivalently, if P is a canonical term the product is taken over all m such that for P, $f_m = 0$. A null Q will be considered equal to 1. The address-function A^- then becomes

$$A^- = \sum_X PQ_P + \prod_{m=0}^{M-1} \bar{y}_m \tag{36}$$

where P is a canonical product of input literals and X indicates that the summation shall include every possible P.

Consider the previous example whose system function S is represented by the truth table of Fig. 15.

Input			Output	
x_2	x_1	x_0	f_1	f_0
0	0	0	1	1
0	0	1	1	0
0	1	0	0	1
0	1	1	1	1
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

Fig. 15. Truth table representation of equation (34).

Apply equation (36) to Fig. 15 and obtain

$$A^- = \bar{x}_2\bar{x}_1\bar{x}_0 + \bar{x}_2\bar{x}_1x_0\bar{y}_0 + \bar{x}_2x_1\bar{x}_0\bar{y}_1 + \bar{x}_2x_1x_0 + x_2\bar{x}_1\bar{x}_0$$
$$+ x_2\bar{x}_1x_0\bar{y}_1 + x_2x_1\bar{x}_0\bar{y}_1\bar{y}_0 + x_2x_1x_0 + \bar{y}_1\bar{y}_0 \qquad (37)$$

Apply the Quine algorithm to equation (37) and find that the prime implicants of A^- are, as before, $\bar{y}_1\bar{y}_0$, $\bar{x}_2\bar{x}_1\bar{y}_0$, $\bar{x}_2x_0\bar{y}_0$, $\bar{x}_2x_1\bar{y}_1$, $\bar{x}_2\bar{x}_0\bar{y}_1$, $x_2\bar{x}_1\bar{y}_1$, $x_2x_0\bar{y}_1$, $\bar{x}_1\bar{x}_0$, and x_1x_0.

Notice that in equation (37) there appears a term $x_2x_1\bar{x}_0\bar{y}_1\bar{y}_0$ that indicates that $x_2x_1\bar{x}_0$ implies no f_m, $0\leq m\leq M-1$. Terms of this type will also appear when applying the Quine algorithm to A^-. The term $\bar{y}_{M-1} \cdots \bar{y}_0$ is included in A^- so that it will be subsumed by these superflous terms and they may immediately be deleted. Another approach is to modify equation (36) so that any term that contains all $\bar{y}_m$, $0\leq m\leq M-1$, is deleted. It is then necessary to modify the Quine algorithm to the same end.

Minimal Irreducible Representations (MIR)

Let w_m be the K digit binary representation for the integer m and K be the smallest integer such that every m in the range $0\leq m\leq M-1$ can be represented by w_m. Introduce the K binary variables a_k, $0\leq k\leq K-1$. Then there exists a canonical product of a and $\bar{a}$ literals, denoted A_m, such that

$$A_m(w_r) = \begin{cases} 1 \text{ if } r = m, \\ 0 \text{ otherwise} \end{cases} \qquad (38)$$

For example, if M = 5, then K = 3 and $A_4 = a_2\bar{a}_1\bar{a}_0$, $A_3 = \bar{a}_2a_1a_0$, $A_2 = \bar{a}_2a_1\bar{a}_0$, $A_1 = \bar{a}_2\bar{a}_1a_0$, and $A_0 = \bar{a}_2\bar{a}_1\bar{a}_0$.

For $0\leq m\leq M-1$, A_m shall be interpreted as the address of the function f_m and for any implicant P of f_m, a product of the form A_mP shall be termed an admissible product and symbolized generically by G. Therefore by equation (38), an admissible product G, is a product such that

$$G(w_r) = A_m(w_r)P = \begin{cases} P \text{ if } r = m, \\ 0 \text{ otherwise} \end{cases} \qquad (39)$$

where P implies f_m, $0\leq m\leq M-1$.

Then a system of functions such as the systems S of equation (31) may be represented by the set of all admissible products associated with it. As an example, consider

$$S: \begin{array}{l} f_2 = x_2\bar{x}_1x_0 + x_1\bar{x}_0 + \bar{x}_2x_0 \\ f_1 = x_1\bar{x}_0 + \bar{x}_2x_0 + \bar{x}_1x_0 \\ f_0 = \bar{x}_1x_0 + x_2\bar{x}_1 \end{array} \tag{40}$$

The system S of equation (40) can be expressed as

$$S: \{a_1a_0x_2\bar{x}_1x_0,\ a_1a_0x_1\bar{x}_0,\ a_1a_0\bar{x}_2x_0,\ \bar{a}_1a_0x_1\bar{x}_0, \\ \bar{a}_1a_0\bar{x}_2x_0,\ \bar{a}_1a_0\bar{x}_1x_0,\ \bar{a}_1\bar{a}_0\bar{x}_1x_0,\ \bar{a}_1\bar{a}_0x_2\bar{x}_1\} \tag{41}$$

which is the set of all admissible products.

Consider another type of House-Rado address-function

$$A^0 = \sum_{m=0}^{M-1} A_m\, f_m \tag{42}$$

or equivalently

$$A^0 = \sum_{\text{all } G \,\varepsilon\, S} G \tag{43}$$

which is the sum of all admissible products of S. By equation (39)

$$A^0(w_m) = f_m \tag{44}$$

for every m, $0 \le m \le M-1$.

Equation (42) expresses the system S of M functions in a single normal form. Suppose that all irreducible representations of A^0 are found, then each representation will express the system S and each f_m, $0 \le m \le M-1$, of every expression of S will necessarily be irreducible. Also consider allowing the set G of A^0 to be created from the set of prime origins of S. Then all irreducible representations of A^0 will indicate expressions of S that are of minimum term count, minimum literal count, and minimum diode count. This occurs because the set of prime origins consists of minimum-literal terms and terms common to one or more f_m of S and the irreducible representations of A^0 will indicate all combinations of sums of these terms representing S. Let all these representations of S be called "Minimal Irreducible Representations" and abbreviated MIR.

The address-function A^0 is a sum of terms and therefore all irreducible representations may be found by applying the House-Rado presence function of equation (27). It is necessary to change the notation of equation (27) to make it explicitly applicable to A^0.

Let there be a set R of H admissible products R_h, $0 \le h \le H-1$, formed from the set of all prime origins of the system S and let wu be a binary

substitution in any R_h, $0 \leq h \leq H-1$, where w is the address substitution and u is the system S independent variable substitution.

Then

$$A^0 = \sum_{h=0}^{H-1} R_h \tag{45}$$

and by equation (27) all MIR of the system S will be indicated by the prime implicants of

$$S(p_0, \ldots, p_{H-1}) = \prod_{\text{all } wu \in A^0 = 1} \{\Delta[R_0(wu), \ldots, R_{H-1}(wu)] + \sum_{h=0}^{H-1} R_h(wu)p_h\} \tag{46}$$

where all $wu \in A^0 = 1$ indicates that the product is to range over all substitutions of wu such that $A^0 = 1$. Since $A^0 = 1$ occurs only when any $f_m = 1$ $0 \leq m \leq M-1$, the product may equivalently be taken over all substitutions of wu for which any $f_m = 1$, $0 \leq m \leq M-1$.

INCOMPLETELY SPECIFIED FUNCTIONS

The multiple-output problem is now modified to include don't cares. Consider a system of functions

$$S^*: \quad f^*_{M-1}, \ldots, f^*_m, \ldots, f^*_0 \tag{47}$$

where each f^*_m, $0 \leq m \leq M-1$, is a function of the independent variables x_{N-1}, $\ldots, x_n, \ldots, x_0$, which are not completely specified. That is, for each m, $0 \leq m \leq M-1$, there is a set D^*_m (perhaps null) of substitutions u, such that $f^*_m(u)$ is defined only for $u \notin D^*_m$. Thus, the don't cares may be different for each f^*_m of S^*. Denote by D^* the class of sets D^*_m, $0 \leq m \leq M-1$.

Then a mod D^* representation of the system S^* is a representation of equation (47) where each f^*_m is a mod D^*_m representation, $0 \leq m \leq M-1$; see equation (24). It also follows that an irreducible mod D^* representation of S^* is a representation of equation (47) where each f^*_m is an irreducible mod D^*_m representation.

It is desired to find all irreducible mod D^* representations of the system S^* that are MIR. As noted earlier the best advantage of the don't care inputs is obtained when they are assumed to cause outputs of value 1.

To attain this end create an auxiliary system

$$S: \quad f_{M-1}, \ldots, f_m, \ldots, f_0 \tag{48}$$

of completely specified functions, by defining

$$f_m(u) = \begin{cases} f^*_m(u) & \text{if } u \notin D^*_m, \\ 1 & \text{if } u \in D^*_m \end{cases} \tag{49}$$

for all m of equation (47). Then equation (46) is applicable to the system S* if the set R of admissible products is formed from the prime origins of the auxiliary system S. Once again, any superfluous term introduced by equation (49) will be dropped as a redundancy when equation (46) is applied.

As before, the completely and incompletely specified systems may be treated equivalently with the proper manipulation of the set of don't cares. Henceforth all systems will be referred to as mod D*, where D* may be null.

MINIMALITY MEASURE

Let each MIR mod D* of the system S* be represented by the set R' of prime origin admissible products, where R' is a subset of R found from a prime implicant of equation (46). Let V' be the set of the P portions (product of x and $\bar{x}$ literals) of R', where V' is a subset of the set of all prime origins of S*, V. Denote by UV' the set of unique products of V'. That is, a single term P may imply several f^*_m of S* and appear in V' more than once, so UV' is the set V' free of these duplications.

For each product P of V', denote by I(P) the number of literals in it and let I'(P) = I(P) except when I(P) = 1, then I'(P) = 0. Let $N_m(V')$ be the number of terms in V' that imply f_m and $N'_m(V') = N_m(V')$ except when $N_m(V') = 1$, then $N'_m(V') = 0$.

Associate with each V' three integers T(V'), L(V'), and D(V'), term count, literal count, and diode count, respectively. Previous definitions yield:

$$T(V') = \text{the number of elements of } UV' \tag{50}$$

the number of unique terms in V'; next

$$L(V') = \sum_{\text{all } P \in UV'} I(P) \tag{51}$$

the number of literals in all unique terms of V'; and

$$D(V') = \sum_{\text{all } P \in UV'} I'(P) + \sum_{m=0}^{M-1} N'_m(V') \tag{52}$$

the number of diodes necessary to construct the representation V' of S* mod D* in a diode AND-to-OR circuit configuration.

Industry has accepted these three measures of minimality for binary switching systems. The particular application of the system and available hardware will dictate which of these three measures or combinations thereof shall be minimized. Therefore a computer is programed to generate all MIR mod D* of S* and calculate T(V'), L(V'), and D(V') for each, leaving the final decision to the investigator.

REVIEW AND EXAMPLE

The general procedure for finding all MIR of the system S* mod D*, where D* can be null, may be stated as follows.

1) Create the auxiliary system S of equation (48) that is defined by equation (49).

2) Form the address-function A^- for the auxiliary system found in 1) using equation (33) or equation (36).

3) Find the prime implicants of A^- using Quine's algorithm and thereby find the set V of prime origins.

4) Discard the auxiliary system S and create the set R of prime origin admissible products from the set V found in 3).

5) Form the address-function A^0 from equation (45) and find the prime implicants of equation (46), where the product of equation (46) shall range over all substitution of wu such that any $f^*_m = 1$ and thereby find all subsets R' of R representing all MIR of the system S* mod D*.

6) Finally, calculate T(V'), L(V'), and D(V') by equations (50), (51), and (52), respectively for each set R' found in 5).

The following real problem, taken from House-Rado (4), will demonstrate this procedure.

Figure 16 lists the specified outputs for given input combinations for the system S*. All other input combinations not listed constitute the set D*. In this problem all D^*_m, $0 \leq m \leq 4$, are the same.

Inputs				Outputs				
x_3	x_2	x_1	x_0	f^*_4	f^*_3	f^*_2	f^*_1	f^*_0
0	0	0	0	1	1	0	0	0
0	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	0	1
0	0	1	1	0	0	1	1	0
0	1	0	0	0	1	0	0	1
1	0	0	0	0	1	0	1	0
1	0	0	1	0	1	1	0	0
1	0	1	0	1	0	0	0	1
1	0	1	1	1	0	0	1	0
1	1	0	0	1	0	1	0	0

Fig. 16. Specified Outputs.

Using equation (49) the auxiliary system is defined in Fig. 17.

Inputs				Outputs				
x_3	x_2	x_1	x_0	f_4	f_3	f_2	f_1	f_0
0	0	0	0	1	1	0	0	0
0	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	0	1
0	0	1	1	0	0	1	1	0
0	1	0	0	0	1	0	0	1
0	1	0	1	1	1	1	1	1
0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	1	1	1
1	0	0	0	0	1	0	1	0
1	0	0	1	0	1	1	0	0
1	0	1	0	1	0	0	0	1
1	0	1	1	1	0	0	1	0
1	1	0	0	1	0	1	0	0
1	1	0	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1

Fig. 17. Auxiliary System.

The address-function $\bar{A}$ is found using equation (36) and is shown in Fig. 18. The function is the sum of the products corresponding to the rows of Fig. 18. For example, the product corresponding to the first entry is $\bar{x}_3\bar{x}_2\bar{x}_1\bar{x}_0\bar{y}_2\bar{y}_1\bar{y}_0$. $\bar{A}$ is the sum of these terms.

* Terms of A^-								
x_3	x_2	x_1	x_0	y_4	y_3	y_2	y_1	y_0
0	0	0	0	-	-	0	0	0
0	0	0	1	0	0	0	-	-
0	0	1	0	0	0	-	0	-
0	0	1	1	0	0	-	-	0
0	1	0	0	0	-	0	0	-
0	1	0	1	-	-	-	-	-
0	1	1	0	-	-	-	-	-
0	1	1	1	-	-	-	-	-
1	0	0	0	0	-	0	-	0
1	0	0	1	0	-	-	0	0
1	0	1	0	-	0	0	0	-
1	0	1	1	-	0	0	-	0
1	1	0	0	-	0	-	0	0
1	1	0	1	-	-	-	-	-
1	1	1	0	-	-	-	-	-
1	1	1	1	-	-	-	-	-

* A 1 in a given position means the variable is not complemented and a 0 means it is complemented; and a - means the variable does not appear.

Fig. 18. The address-function A^-.

* Prime Implicants								
x_3	x_2	x_1	x_0	y_4	y_3	y_2	y_1	y_0
1	-	1	-	-	0	0	0	0
1	-	1	1	-	0	0	-	0
1	1	-	-	-	0	-	0	0
-	-	1	1	0	0	0	-	0
-	-	1	0	0	0	0	0	0
1	0	0	-	0	-	0	0	0
-	0	0	0	0	-	0	0	0
1	0	0	0	0	-	0	-	0
0	-	1	-	0	0	-	0	0
0	-	-	1	0	0	0	-	0
0	-	0	0	0	-	0	0	0
0	-	0	1	0	0	0	-	-
0	-	1	1	0	0	-	-	0
1	-	0	1	0	-	-	0	0
0	1	-	-	0	-	0	0	-
0	-	1	0	0	0	-	0	-
1	-	1	0	-	0	0	0	-
0	0	0	0	-	-	0	0	0
-	1	-	1	-	-	-	-	-
-	1	1	-	-	-	-	-	-
-	-	-	-	0	0	0	0	0

* See footnote to Fig. 18.

Fig. 19. The prime implicants of A^-.

* Admissible Products						
x_3	x_2	x_1	x_0	a_2	a_1	a_0
1	-	1	-	1	0	0
1	-	1	1	1	0	0
1	-	1	1	0	0	1
1	1	-	-	1	0	0
1	1	-	-	0	1	0
-	-	1	1	0	0	1
-	-	1	0	0	0	0
1	0	0	-	0	1	1
-	0	0	0	0	1	1
1	0	0	0	0	1	1
1	0	0	0	0	0	1
0	-	1	1	0	1	0
0	-	-	-	0	0	1
0	-	0	0	0	1	1
0	-	0	1	0	0	1
0	-	0	1	0	0	0
0	-	1	1	0	1	0
0	-	1	1	0	0	1
1	-	0	1	0	1	1
1	-	0	1	0	1	0
0	1	-	-	0	1	1
0	1	-	-	0	0	0
0	-	1	0	0	1	0
0	-	1	0	0	0	0
1	-	1	0	1	0	0
1	-	1	0	0	0	0
0	0	0	0	1	0	0
0	0	0	0	0	1	1
-	1	-	1	1	0	0
-	1	-	1	0	1	1
-	1	-	1	0	1	0
-	1	-	1	0	0	1
-	1	-	1	0	0	0
-	1	1	-	1	0	0
-	1	1	-	0	1	1
-	1	1	-	0	1	0
-	1	1	-	0	0	1
-	1	1	-	0	0	0

* See footnote to Fig. 18.

Fig. 20. The set R of prime origin admissible products.

The prime implicants of A^- are found by the Quine algorithm and listed in Fig. 19. The P portions of these terms constitute the set V of prime origins.

The set R of prime origin admissible products is found from Fig. 19 and listed in Fig. 20.

The sum of the terms of Fig. 20 constitutes the address-function A^0. Find all irreducible representation of A^0 as the prime implicants of equation (46). Thus obtain all the subsets R' of R and all MIR of Fig. 17. For this system there are 224 MIR.

One of these representations has a minimum D(V') of 42 diodes and is:

$$f_0 = \bar{x}_3\bar{x}_1x_0 + \bar{x}_3x_2 + \bar{x}_3x_1\bar{x}_0 + x_3x_1\bar{x}_0$$
$$f_1 = x_3\bar{x}_2\bar{x}_1\bar{x}_0 + x_3x_1x_0 + \bar{x}_3\bar{x}_1x_0 + \bar{x}_3x_1x_0$$
$$f_2 = x_3x_2 + x_3\bar{x}_1x_0 + \bar{x}_3x_1x_0 + \bar{x}_3x_1\bar{x}_0$$
$$f_3 = x_3\bar{x}_2\bar{x}_1\bar{x}_0 + x_3\bar{x}_1x_0 + x_3x_2 + \bar{x}_3\bar{x}_2\bar{x}_1\bar{x}_0$$
$$f_4 = x_3x_2 + \bar{x}_3\bar{x}_2\bar{x}_1\bar{x}_0 + x_3x_1x_0 + x_3x_1\bar{x}_0$$

It is noted by House-Rado (4) that this problem has been studied by a design engineer who relied solely upon his educated intuition. His representation required 44 diodes; this is within 5 percent of the optimal design.

SUMMARY

This report compiles the House-Rado algorithms for the reduction of Boolean expressions on a binary digital computer. It may be divided into two major sections. The first section deals with the subject of multiple input-single output switching functions and develops the House-Rado presence function, which is an algorithm for finding all irreducible representations of any single-output switching system. Also Quine's algorithm for finding prime implicants is developed and ground work is laid for the second section.

The second section deals with the major portion of the House-Rado work, that of multiple-output switching systems. The address-function A^- is developed first; A^- is an algorithm for finding the prime origins of a system of functions representing a multiple-output switching network. Next the address function A^0 is developed; A^0 is an algorithm for finding all

irreducible representations of a system of functions. Finally the concept of minimal irreducible representations (MIR) is developed by introducing the prime origins found from A^{-} into the address-function A^{0}.

In both sections these concepts are developed for completely specified functions and then it is shown that with the proper manipulation of the don't care inputs the incompletely specified functions can be handled equivalently.

The report concludes with a real design problem whose solution demonstrates the usefulness of these algorithms.

ACKNOWLEDGMENT

This author wishes to thank Dr. Charles A. Halijak for his counsel and encouragement throughout the preparation of this report.

REFERENCES

1. Bartee, T.C.
Computer Design of Multiple-Output Logical Networks. IRE Trans. on Electronic Computers, Vol. EC-10, No. 1, p. 21, March, 1961.

2. Battelle Memorial Institute.
The Formulation of Automatic Checkout Techniques. ASD Tech. Documentary Rept. ASD-TDR-62-291, AF Contract No. 33 (616) - 7761. March, 1962.

3. Ghazala, M.J.
Irredundant Disjunctive and Conjunctive Forms of a Boolean Function. IBM Journal, p. 171. April, 1957.

4. House, R.W. and Rado, T.
On a Computer Program for Obtaining Irreducible Representations for Two-Level Multiple Input-Output Logical Systems. J. ACM, Vol. 10, No. 1, p. 48. January, 1961.

5. House, R.W. and Rado, T.
Implementation of Logic. IRE Trans. On Military Electronics, Vol. MIL-6, No. 3. July, 1962.

6. House, R.W. and Rado, T.
A Reduction Algorithm for Incompletely Specified Functions. Unpublished.

7. Quine, W.V.
On Core and Prime Implicants of Truth Functions. American Mathematical Monthly, Vol. 66, p. 755. November, 1959.

8. Rado, T.
Comments on the Presence Function of Gazalé. IBM Journal, p. 268. April, 1962.

9. Whitesitt, J.E.
Boolean Algebra and It's Applications. Massachusetts: Addison-Westley Company, Inc., 1961.

COMPUTER REDUCTION OF BOOLEAN EXPRESSIONS

by

DAVID GORDON DUTRA

B. S. E. E., University of the Pacific
Stockton, California, 1962

AN ABSTRACT OF
A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1965

The purpose of this report is to compile in an orderly manner the work of R.W. House and T. Rado on the reduction of Boolean switching functions. Mathematical rigor is de-emphasized to provide the reader with insight to the underlying concepts.

The development begins by discussing single dependent variable Boolean functions of a set of completely specified independent variables. The concept of prime implicants is introduced and Quine's algorithm for finding prime implicants is developed. Next, irreducible representations are introduced and an algorithm is developed for finding all irreducible representations of a function. These concepts are then extended to incompletely specified functions and it is shown that when irreducible representations are sums of prime implicants they are minimal representations. This section concludes with an example in order to unify these ideas.

The next section parallels the first with a discussion of systems of functions representing multiple-output switching networks. The topic of prime origins is introduced and an address function (A^-) algorithm is developed for calculating prime origins. This is followed by a second address-function (A^0) algorithm for calculating irreducible representations of a systems of functions. Once again, it is shown that by introducing prime origins into the address-function A^0 minimal irreducible representations of the system result. There can be many of these irreducible representations, on the order of several hundred, so a minimality measure is assigned to each representation; these are literal count, term count, and diode count.

These ideas are then unified by presenting a real problem that House and Rado solved on a digital computer. It is shown that there are 224 minimal irreducible representations of this system; the diode-optimal representation is presented.