Electronics Research Laboratory
University of California
Berkeley, California

TOPICS IN THE USE OF LINEAR PROGRAMMING IN THE MINIMIZATION OF BOOLEAN FUNCTIONS

by

M. A. Breuer

Supported in part by the Air Force Office of Scientific Research under Contract AFOSR-62-156.

December 1, 1962

ACKNOWLEDGMENT

The author wishes to thank Professors Harry D. Huskey and E. Eisenberg for their helpful suggestions and constructive criticism offered during the preparation of this material.

M. A. B.

ABSTRACT

This report deals with topics related to the use of linear programming in the minimization of Boolean functions [1].

(1) An algorithm is presented for determining the optimal factors in both "AND-AND-OR" and restricted "AND-OR-AND-OR" gating.

(2) A procedure has been formulated for taking into consideration "don't care" conditions in the minimization process.

(3) A method is given for simplifying functions in maxterm form, without converting these functions to minterm form.

(4) An algorithm is presented for determining all the factors a set of functions may have in common. These common factors are then introduced into the linear program in order to obtain the optimal minimal cost form.

(5) The dual of the original linear program is derived, and some pertinent theorems and physical properties are discussed.

(6) The quadratic program, derived from letting the fan-in factor be a variable, is discussed.

DEFINITIONS AND SYMBOLS

1.) Linear Program:

a) objective function: $\sum_{j=1}^{n} C_j P_j' = Z \text{ (min.)}$

b) constraint inequalities: $\sum_{j=1}^{n} a_{ij} P_j' \geq b_i \quad i = 1, 2, \ldots, m$

c) non-negative variables: $P_j' \geq 0 \quad j = 1, 2, \ldots, n$

The C_j's are positive cost coefficients.

2.) n_X: Boolean variable

3.) X and $\overline{X}$: Literals contained in n_X.

4.) ℓ_i: General literal

5.) P_j: Boolean product (prime implicant) $P_j = \prod_i \ell_i$

6.) B_j: Generalized Boolean product, defined where used.

7.) F_k: k'th Boolean function $F_k = \sum_j B_j$ or $F_k = \prod_i \overline{B}_i$

8.) G_k: Minimal form of F_k

9.) $B_j' \in (0, 1)$: Bivalued variable, where

$B_j' = 1 \Longrightarrow B_j$ is a term in G_k

$B_j' = 0 \Longrightarrow B_j$ is not a term in G_k

10.) $\supset$: Contained in.

11.) $\cap$: Intersection.

TABLE OF CONTENTS

I. INTRODUCTION

In Part I of this report [1], a method was presented for minimizing Boolean functions, by employing linear programming. The classical model of a Boolean function in vector space was presented. Each product term was shown to be equivalent to a set of vertices in the vector space. From this model, the linear program was formulated for minimizing the Boolean functions in prime implicant form with arbitrary constant cost coefficients. Variable nonlinear cost functions, derived from the fan-out problem, were then considered, as well as lead length, loading and fan-in restrictions.

To supplement this work, the following topics, dealing with the use of linear programming in the simplification of Boolean functions, have been investigated.

1. Factorization: The restriction to "AND-OR" logic has been removed. "AND-AND-OR" and restricted "AND-OR-AND-OR" gating is now possible. An algorithm is given for determining the optimal choice of factors for minimal cost.

2. Redundancies: A procedure has been formulated for taking into consideration "don't care" conditions in the minimization process.

3. Maxterms: Two analogous methods have been developed for simplifying Boolean functions in maxterm form using linear programming. The second procedure is unique in that it is based entirely on the maxterm representation.

4. Common Factors: When many functions are minimized together, and common factors are used, the minimal cost form need not contain only prime implicants. An algorithm is given for determining all product terms to be considered in the minimal form.

5. Duality: The dual of the linear program used in the minimization process has been analyzed, and additional insight to the simplification process has been obtained. Theorems, pertaining to the simplifications of primal and dual linear programs prior to solution, have been derived.

6. Fan-in considerations: The problem of letting the fan-in factor be a variable has been attacked. The objective is to have the program determine the optimal value for the maximum number of inputs to an "OR" gate.

It is hoped that this work will be a step forward in automatic logical design and simplification.

II. FACTORIZATION

In Part I of this report [1], only two level "AND-OR" gating was considered. A procedure is now presented for determining the optimal factorization of product and sum terms from the prime implicants. Both "AND-AND-OR" and restricted "AND-OR-AND-OR" gating will be discussed. Two important motivations for factoring are a realization of greater savings in the cost of implementation, and a solution to the fan-out problem. The fan-out factor F_{out} is defined to be the maximum number of inputs to an "AND" gate.

A few basic definitions will now be given, after which the linear program formulation of the Boolean simplification process will be given.

A factor is defined as $f_a = \ell_a \cdots \ell_e$, where $f_a \supseteq P_j$ for one or more j. Recall that the P_j's are prime implicants, and the ℓ's are literals. A general product term, which need not be a prime implicant, will be represented by $B_r (B_r = \prod_i \ell_i)$. The two types of factorization will be discussed separately.

A. "AND-AND-OR" logic:

In implementing P_j, a sub-set of its factors are first formed in separate "AND" gates, and the results are "ANDED" together along with any other terms of P_j which were not explicitly considered as factors. The problem is to determine the correct sub-set of factors for each P_j so that the total cost of implementing the logic is minimal.

Lemma 1:

If $f_\alpha \supset f_\beta \supseteq P_j$, and if all costs are positive, then f_α and f_β are not both in the minimal cost implementation for P_j.

Proof: Since $f_\alpha \supset f_\beta$, $(f_\beta = 1) \Rightarrow (f_\alpha = 1)$

Since $f_\beta \supseteq P_j$, $(f_\beta = 0) \Rightarrow (P_j = 0)$

If f_β is used in P_j, f_α is redundant with a positive cost, and hence can not be in the minimal cost implementation.

If the factor f_α exists in m different prime implicants, where $m > F_o'$, and where F_0' is the maximum number of times a factor may be used without the necessity of an amplifier, then three different implementations are possible. First, an amplifier may be used. Secondly, the factor may be formed more than once. Finally, f_α may be used as a factor in $m' = F_0'$ of the prime implicants, and no factor used in the remaining $m - m'$ prime implicants. This will usually be the case when $m - m' = 1$, since it does not pay to use a factor only once, unless it is used for fan-out purposes.

The general algorithm for the linear programming formulation is as follows.

1. From the prime implicant representation of the function to be simplified, determine a set of product terms to be considered as possible factors in the minimal form of the function. This set need not be the entire set of product terms available as factors.

2. For each prime implicant P_j in F, determine a set of product terms $\{P_j\}$, each of whose elements B_r are logically equivalent to P_j, but which contain a different combination of factors. All combinations of factors need not be considered. To each element B_r in $\{P_j\}$, assign a new variable $B_r' \in (0, 1)$.

3. Determine the constraint inequalities in terms of the original prime implicants P_j'.

4. For each P_j' in step 3, substitute the set of product terms $\{P_j'\}$ determined in step 2.

Due to Lemma 1, each inequality in step 3 containing only one entry may be represented as an equality in step 4. Note that the inequality in step 3 could have been changed to an equality, since

$$P_j' \geq 1 \Longrightarrow P_j' = 1 \quad \text{for } C_j > 0 .$$

5. Construct the objective function by determining the cost for each B_r. The cost of implementing each factor must also be included. The cost associated with factor f_a is C_{f_a} if f_a is used, and is zero otherwise. To formulate this statement into the linear program (L. P.), let

$$N_{f_a} = \sum_r B_r' \qquad a = 1, 2, \ldots$$

where r is summed over all G_r containing f_a.

Now, if $N_{f_a} > 0$ let $\delta_a = 1$

if $N_{f_a} < 0$ let $\delta_a = 0$

Consider the inequality

$$N_{f_a} \leq \delta_a M_a \tag{1}$$

where $M_a = N_{f_a}$ (Max.)

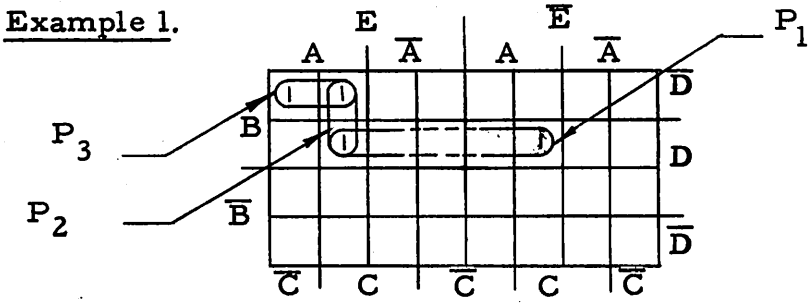
For $\delta = 0$, (1) gives $N_{f_a} \leq 0$

and for $\delta = 1$, (1) gives $N_{f_a} \leq M_a$

Therefore, δC_f is the cost associated with implementing the factor f_a. In the actual L. P., N_{f_a} is replaced by $\sum_r B'_r$.

Integer linear programming must be used to insure $\delta_a \in (0,1)$ If N_{f_a} (Max.) $> F_o$, the cost of an amplifier or additional gate for forming the factor more than once must be accounted for.

Example 1.



$$F = \underbrace{ABCD}_{P_1} + \underbrace{ABCE}_{P_2} + \underbrace{AB\overline{D}E}_{P_3}$$

1. The only factors to be considered are

$$f_1 = ABC \text{ and } f_2 = AB$$

2. A factored term in a prime implicant will be indicated by brackets (factor).

$$\{P_1\} = \{\underbrace{ABCD}_{B_1}, \underbrace{(ABC)D}_{B_4}, \underbrace{(AB)CD}_{B_5}\}$$

$$\{P_2\} = \{\underbrace{ABCE}_{B_2}, \underbrace{(ABC)E}_{B_6}, \underbrace{(AB)CE}_{B_7}\}$$

$$\{P_3\} = \{\underbrace{AB\overline{D}E}_{B_3}, \underbrace{(AB)\overline{D}E}_{B_8}\}$$

3. The original constraint inequalities are

$$P_1' \geq 1$$

$$P_1' + P_2' \geq 1$$

$$P_2' + P_3' \geq 1$$

$$P_3' \geq 1$$

4. In terms of the new variables, introduced because of factoring, the constraints become

$$B_1' + B_4' + B_5' \geq 1$$

$$B_1' + B_2' + B_4' + B_5' + B_6' + B_7' \geq 1$$

$$B_2' + B_3' + B_6' + B_7' + B_8' \geq 1$$

$$B_3' + B_8' \geq 1$$

5. Assuming unit diode costs for each term in an "AND" or an "OR" gate, the objective function is

$$5B_1' + 5B_2' + 5B_3' + 3B_4' + 4B_5' + 3B_6' + 4B_7' + 4B_8' + 3\delta_{f_1} + 2\delta_{f_2} = Z(\text{Min.})$$

The additional constraint inequalities required for δ_{f_1} and δ_{f_2} are

$$N_{f_1} = B_4' + B_6' \leq 2\delta_{f_1} \qquad N_{f_2} = B_5' + B_7' + B_8' \leq 3\delta_{f_2}$$

The solution is

$$B_5' = B_8' = \delta_{f_2} = 1$$

$$B_1' = B_2' = B_3' = B_4' = B_6' = B_7' = \delta_{f_1} = 0$$

min. $Z = 10$

It is important to note that the constraint inequalities with the factored variables B_r' were written from the original prime implicant linear program having variables P_j'. A second approach would be to minimize the Boolean function in terms of the original prime implicants P_j, and then factor the resulting equation. These two methods are not equivalent, and only the former guarantees the minimum cost function. The reason for this is that if a choice must be made between P_j and P_i, where $C_j > C_i$, but where the cost of implementing the function in factored form is less when P_j is present than if P_i were present, then the latter method does not give the optimal solution.

B. "AND-OR-AND-OR" Logic:

First it is necessary to extend the definition of a general product term B_r, where now one of the elements, rather than being a literal, may be a sum of products. The general form of B_r is then $B_r = \underbrace{(\ell_{a_1} \ell_{a_2} \cdots \ell_{a_i})}_{\text{factor}} [\ell_{b_1} \ell_{b_2} \cdots \ell_{b_j} + \ldots + \ell_{e_1} \ell_{e_2} \cdots \ell_{e_k}]$.

B_r is then a restricted form of "AND-OR-AND" logic, and F is the "OR" of B_r terms, i.e., $F = \sum_r B_r$. The procedure for formulating the linear program is as follows:

1. From the prime implicants P_j determine the set of factors to be considered.

2. Determine a new set of product terms B_r to be considered for the minimal form of F. These terms are formed by factoring various combinations of the prime implicants, using the set of factors derived in step 1. To each product term B_r in F, assign a variable $B_r' \epsilon (0, 1)$. Construct a list of logical implications, indicating which of the new set of product terms B_r is true as a consequence (direct implication) of one of the original product terms P_j being true. The elements of this lists from the set $\{P_j\}$.

3. Determine the constraint inequalities in terms of the original P'_j.

4. For each P'_j in step 3, substitute the set of new variables B'_r derived from the logical implications obtained in step 2. A variable B'_r may appear at most just once in any one inequality.

5. Construct the objective function. In the past, we have associated with the total cost of each product term B_r, the additional cost of one diode required to "OR" this term to the remaining terms in F. However, with "AND-OR-AND-OR" gating, it may turn out that there is only one term in F. For this case, we must subtract from the objective function the cost of one diode. Let this cost be $c_d \delta_d - c_d$ where

$\sum_r B'_r \leq 1 + \delta_d M$, $M = (\sum_r B'_r)_{Max.}$ and $\delta_d \in (0, 1)$. For

$\sum_r B'_r \geq 2$, $\delta_d = 1$; for $\sum_r B'_r = 1$, $\delta_d = 0$; and $\sum_r B'_r$ cannot equal zero.

Example 2.

The function given in Example 1 will again be considered.

$$F = \underbrace{ABCD}_{P_1} + \underbrace{ABCE}_{P_2} + \underbrace{AB\bar{D}E}_{P_3}$$

1. The only factors to be considered are $f_1 = ABC$, $f_2 = AB$

2. The new product terms to be considered are

$$\underbrace{ABCD}_{B_1}, \quad \underbrace{ABCE}_{B_2}, \quad \underbrace{AB\bar{D}E}_{B_3}, \quad \underbrace{(ABC)\,[D+E]}_{B_4}$$

$$\underbrace{(AB)\,[CD + CE + \overline{D}E]}_{B_5},\quad \underbrace{(AB)\,[CD + CE]}_{B_6}$$

$$\underbrace{(AB)\,[CE + \overline{D}E]}_{B_7},\quad \underbrace{(AB)\,[\overline{D}E + CD]}_{B_8}$$

Now

$$P_1 = 1 \Longrightarrow \begin{cases} B_1 = 1 \\ B_4 = 1 \\ B_5 = 1 \\ B_6 = 1 \\ B_8 = 1 \end{cases}$$

Hence $\{P_1\} = \{B_1, B_4, B_5, B_6, B_8\}$

Also, in similar fashion we obtain:

$$\{P_2\} = \{B_2, B_4, B_5, B_6, B_7\}$$

$$\{P_3\} = \{B_3, B_5, B_7, B_8\}$$

3. The original constraint inequalities are

$$\begin{aligned} P_1' &\geq 1 \\ P_1' + P_2' &\geq 1 \\ P_2' + P_3' &\geq 1 \\ P_3' &\geq 1 \end{aligned}$$

4. The new constraints are determined from the logical implications.

$$
\begin{aligned}
&B_1' \qquad\qquad + B_4' + B_5' + B_6' \qquad + B_8' \geq 1\\
&B_1' + B_2' \qquad + B_4' + B_5' + B_6' + B_7' + B_8' \geq 1\\
&\quad B_2' + B_3' + B_4' + B_5' + B_6' + B_7' + B_8' \geq 1\\
&\qquad B_3' \qquad + B_5' \qquad + B_7' + B_8' \geq 1
\end{aligned}
$$

5. The objective function is, assuming equal diode costs,

$$5B_1' + 5B_2' + 5B_3' + 7B_4' + 14B_5' + 11B_6' + 11B_7' + 11B_8' + \delta_d =$$

$$= Z\,(\text{min.}) + 1$$

where $B_1' + B_2' + B_3' + B_4' + B_5' + B_6' + B_7' + B_8' + 8\delta_d \leq 1$

The two optimal solutions are $\delta_d = B_1' = B_3' = 1$, $B_2' = B_4'$

$$= B_5' = \ldots = B_8' = 0 \quad \text{and} \quad B_8' = 1,\ B_1'$$

$$= B_2' = \ldots = B_7' = \delta_d = 0\ ; \quad \text{min.}\ \ Z = 10\ .$$

The case of using (AB) as a factor in B_5, B_6, B_7, and B_8 has not been considered. However, this case could have been handled by employing the procedure of part A of this section.

Summary

A simple though somewhat cumbersome procedure for "AND-AND-OR" and restricted-"AND-OR-AND-OR" factorization has been presented. The technique can be extended for other types of gating sequences. The advantages of factoring are a realization of greater savings in the cost of implementation, and a solution to the fan-out problem.

There are two main drawbacks to these procedures. First, the designer must determine all combinations of factors to be considered, and then formulate the problem. If a great number of factors are possible, this becomes a large, time consuming job. This problem can be easily solved by having the computer formulate the linear program directly from a given set of Boolean functions, preferably not even in prime implicant form.

The second problem stems from the fact that with a large number of factors, the number of variables B_j' in the linear program increases quite rapidly. The result is that only a few Boolean functions can be considered at any one time. However, this is the opposite effect desired. The difficulty can be rectified by employing linear programming routines which can accommodate a large number (1000) of variables.

III. REDUNDANCIES

The case where specific combinations of the variables are redundant, and hence represent "don't care" conditions, is now considered. Redundancies in this context refer to the fact that certain combinations of states of the system being described by the Boolean functions can not occur. Each combination of states of the system corresponds to a vertex in our n-dimensional space. In order to take into account these "don't care" conditions when simplifying the Boolean function F_k, a new set of prime implicants $\{\tilde{P}\} = \{\tilde{P}_1, \ldots \tilde{P}_j, \ldots, \tilde{P}_m\}$ must be determined. The $\tilde{P}_j$'s are determined in the standard manner, where now the "don't care" vertices are included along with the vertices of F_k. The constraint inequalities for the linear program are as follows:

$\sum_{j=1}^{m} a_{v_j} \tilde{P}'_j \geq 1$ for all v in F_k. An inequality exists only for the vertices in F_k. If a $\tilde{P}_j$ contains only "don't care" vertices, then $\tilde{P}_j$ will not appear in the L. P.

Example 3.

Consider function F_1 of Example 1 (page 10) of [1], where we now include the "don't care" term $\overline{A}BD$. This implies that the states represented by the combinations ($\overline{A}$, B, C, D) and ($\overline{A}$, B $\overline{C}$, D) can not occur. The function is shown in Figure 1.

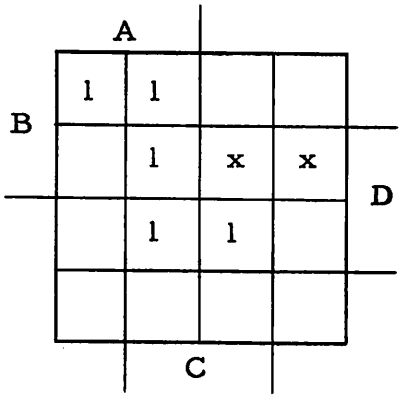


Figure 1. F_1 and "don't care" conditions

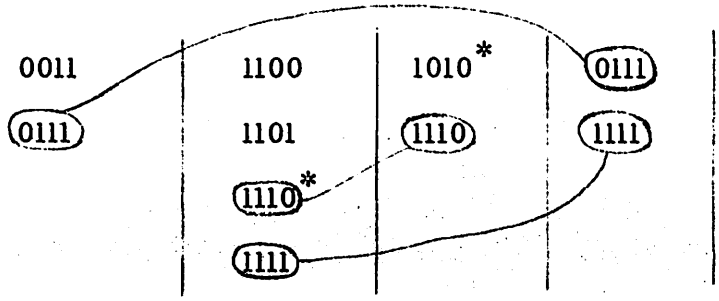
The new prime implicants are

$\tilde{P}_1$	$\tilde{P}_2$	$\tilde{P}_3$	$\tilde{P}_4$
$AB\overline{D}$	CD	$\overline{A}BD$	ABC

The cells are

C_1	C_2	C_3	C_4
(0, -, 1, 1)	(1, 1, -, -)	(1, -, 1, 0)	(-, 1, 1, 1)

The vertices are



*--An asterisk indicates "don't care" vertices. $\tilde{P}_3$ may be neglected, since it contains only "don't care" vertices. The logical constraint inequalities are

$$\tilde{P}_1' \geq 1 \text{ for vertex (0011)}$$

$$\tilde{P}_1' \qquad \tilde{P}_4' \geq 1 \text{ " " (0111)}$$

$$\tilde{P}_2' + \tilde{P}_4' \geq 1 \text{ " " (1111)}$$

$$\tilde{P}_2' \geq 1 \text{ " " (1100)}$$

$$\tilde{P}_2' \geq 1 \text{ " " (1101)}$$

The solution is $G_1 = \tilde{P}_1 + \tilde{P}_2 = AB\overline{D} + CD$ irrespective of the objective function.

IV. MAXTERM REPRESENTATION

In the previous analysis, it was assumed that the Boolean function being minimized was expressed as a sum of product terms. That is, $F = \sum_j P_j$. A function expressible in this form is said to be in minterm form, where each product term P_j is

called a minterm.

$$P_j = \prod_i \ell_i \quad \text{where } \ell_i \text{ is the } i^{th} \text{ literal.}$$

The same function can also be expressed as a product of sum terms.

$$F = \sum_k \overline{P}_k \quad \text{where } \overline{P}_k = \sum_m \ell_m$$

and where the k's take on a different set of values from the j's. The function expressed as such is said to be in maxterm form, and a technique for determining the minimal maxterm representation is given in Phister [2].

In order to realize a savings in cost, it is sometimes desirable to implement a function in maxterm form rather than minterm form. For example, consider the function

$$F = AB + BC + \overline{A}\overline{B} = AB + \overline{A}C + \overline{A}\overline{B} \text{ (9 diodes).}$$

Now $\overline{F} = \overline{A}B\overline{C} + A\overline{B}$, hence

$$F = (A + \overline{B} + C)(\overline{A} + B) \text{ (7 diodes)}$$

One method for formulating the minimization process as a linear program, for a function F in maxterm form, consists of the following two steps.

1) Express $\overline{F}$ in minterm form. $\overline{F} = \sum_j P_j$

2) Minimize $\overline{F}$ using linear programming techniques, where the C_j's in the objective function are the costs associated with implementing $\overline{P}_j$ rather than P_j. The minimal form of F is $G = \prod_j \overline{P}_j$ where j is summed over a sub-class of the original set of j's, and where in the solution of the linear program,

$P'_j = \Longrightarrow \overline{P}_j$ is in G, and $P'_j = 0 \Longrightarrow \overline{P}_j$ is not in G.

(Note if $P_j = A\overline{B}$, then $\overline{P}_j$ $(\overline{A} + B)$.

It is also possible to formulate the linear program directly from the maxterm representation of the function to be minimized.

Let $F = \prod_{k=1} \overline{P}_k$ where $\overline{P}_k = \sum_{m=1}^{M_k} \ell_{k_m}$.

Associate with each $\overline{P}_k$ a set of vertices $\{V_k\}$ consisting of the intersection of all the vertices not contained in ℓ_{k_m}, $m = 1, \ldots, M_k$. $\{V_k\} = \{\underline{\nu}_{k_1} \cap \overline{\nu}_{k_2} \cap \ldots \cap \overline{\nu}_{k_m} \cap \ldots \cap \overline{\nu}_{k_{M_k}}\}$ where $\overline{\nu}_{k_m}$ is the set of vertices not contained in ℓ_{k_m}.

Note that $\{V_k\}$ is the set of vertices contained in P_k*. Hence, if any vertex in $\{V_k\}$ is true, $\overline{P}_k$ is false.

Let $a_{v_k} = 1$ if $\{V_k\}$ contains vertex v.

Let $a_{v_k} = 0$ if $\{V_k\}$ does not contain vertex v.

The deletion of a product term from the function F is equivalent to replacing that term by one. Hence the new function will always be true whenever the original function is true. The criterion which must be satisfied for the simplified function G, is that G must be false whenever the original function F is false. Expressing this condition as a set of constraint inequalities, then $\sum_{k=1}^{K} a_{v_k} \overline{P}'_k \geq 1$ for all v not in F. The objective function is

$$\sum_{k=1}^{K} C_k \overline{P}'_k = Z(\text{Min.})$$

---

*In general, if $\{V_k\}$ is empty, then $P_k = 0$, and $\overline{P}_k = 1$, e.g. $\overline{P}_k = A + \overline{A}$. If $\{V_k\}$ contains all of the vertices in the space, then $P_k = 1$, and $\overline{P}_k = 0$.

The formulation for the maxterm problem is essentially the same mathematically as that for the minterm problem. The main difference is the statement of the criterion used in defining the properties of the minimal form G.

Example 4.

Consider the function containing the true vertices $A\overline{B}\overline{C}$, $A\overline{B}C$, and $\overline{A}B\overline{C}$, as shown in Figure 2.

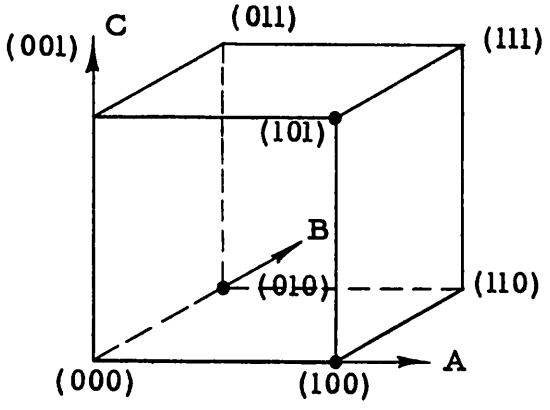


Figure 2. Function F

$$F = \underbrace{(\overline{A} + \overline{B})}_{\overline{P}_1}\ \underbrace{(\overline{B} + \overline{C})}_{\overline{P}_2}\ \underbrace{(\overline{A} + \overline{C})}_{\overline{P}_3}\ \underbrace{(A + B)}_{\overline{P}_4}$$

$\overline{\nu}_{\overline{A}}$ = (100), (101), (111), (110)

$\overline{\nu}_{\overline{B}}$ = (010), (011), (111), (110)

Therefore $\{V_1\} = \{\overline{\nu}_{\overline{A}} \cap \overline{\nu}_{\overline{B}}\}$ = { (111), (110) }

Note (111) and (110) are the vertices contained in $P_1 = AB$.

Also,

$$\{V_2\} = \{(011),\quad (111)\}$$

$$\{V_3\} = \{(001),\quad (011)\}$$

$$\{V_4\} = \{(000),\quad (001)\}$$

Since G must be false whenever F is false, then

$$\begin{aligned}
&\overline{P}'_1 && && \geq 1 && \text{for vertex (110)}\\
&\overline{P}'_1 + \overline{P}'_2 && && \geq 1 && \text{for vertex (111)}\\
&\overline{P}'_2 + \overline{P}'_3 && && \geq 1 && \text{for vertex (011)}\\
&\overline{P}'_3 + \overline{P}'_4 && && \geq 1 && \text{for vertex (001)}\\
&\overline{P}'_4 && && \geq 1 && \text{for vertex (000)}
\end{aligned}$$

V. COMMON FACTORS

In this section, only two level "AND-OR" logic will be considered. A common factor is defined as a product term $B_r = \prod_i \ell_i$ which is used as an "OR" term in two or more functions.

Classical simplification procedures usually begin with the function in prime implicant form. Since the cost associated with each product term is positive, the results of these procedures do give the optimal solution in terms of minimal cost, when only one function is being considered. However, if more than one function is being simplified, and common factors are allowed, then the restriction to prime implicants will not always permit a true overall optimal solution. For example, consider the carry function

$$F_1 = AB\overline{C} + A\overline{B}C + \overline{A}BC + ABC \quad \text{(16 diodes)}$$

In prime implicant form, we have

$$G_1 = AB + BC + AC \qquad \text{(9 diodes)}$$

Now

$$AC = ABC + A\overline{B}C$$

Assume that $A\overline{B}C$ is an essential prime implicant in function G_2. Hence, $A\overline{B}C$ can be used in G_1 at the cost of one diode required in the "OR" gate. Then

$$G_1^* = AB + BC + ABC + A\overline{B}C$$
$$G_1^* = AB + BC + A\overline{B}C \qquad \text{(7 diodes)}$$

A reduction in cost is therefore possible if the restriction that G_k consists only of a subset of the prime implicants of F_k is removed. Note also that each product term P_j need not be a prime implicant of some function. As an example, consider the functions F_2 and F_3 shown below, in Figure 3.

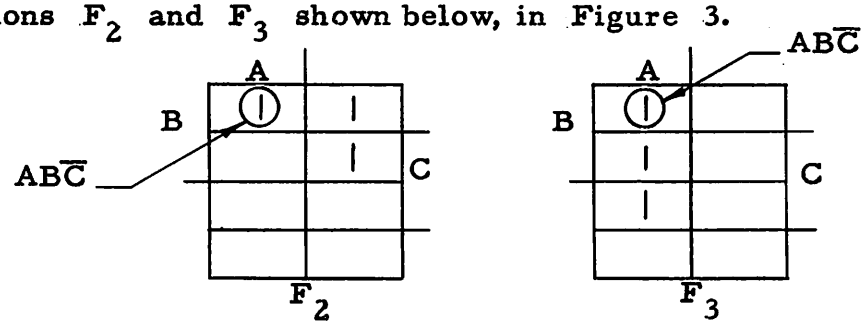


Figure 3. Functions F_2 and F_3

$F_2 = \overline{A}B + B\overline{C}$ (6 diodes) $\qquad F_3 = AB + AC$ (6 diodes)

Total cost is 12 diodes.

Using $AB\overline{C}$, which is not a prime implicant, as a common factor

$$B_1 = AB\overline{C} \qquad \text{(3 diodes)}$$

$$F_2 = \overline{A}B + B_1 \text{ (4 diodes)} \qquad F_3 = AC + B_1 \text{ (4 diodes)}$$

The total cost is 11 diodes.

Finally, it is possible that G_k may contain a redundant prime implicant. For example, consider the function F_5 shown in Figure 4.

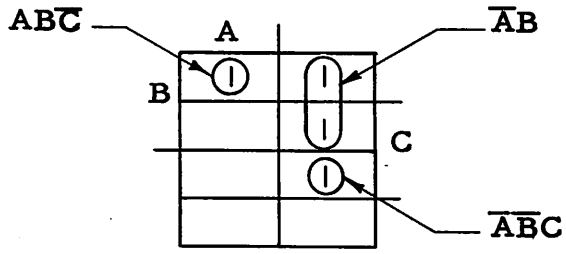


Figure 4. F_5

$$F_5 = AB\bar{C} + \bar{A}BC + \bar{A}B\bar{C} + \bar{A}\bar{B}C$$

If $AB\bar{C}$ and $\bar{A}\bar{B}C$ have been formed for use in some other functions, then the minimal cost form of F_5 may be

$$G_5 = AB\bar{C} + \bar{A}B + \bar{A}\bar{B}C$$

where $\bar{A}B$ is a redundant prime implicant.

A procedure for determining the true minimal form for implementing many Boolean functions, where common factors are allowed, is now given. The main part of this method deals with determining the set of allowable common factors. Linear programming will be used as a tool for finding the solution. The algorithm for formulating the linear program, where all possible representations of the Boolean functions are considered, consists of six parts.

1. Determine all of the prime implicants $P_j (j = 1, 2, \ldots, m)$ contained in the functions $F_k (k = 1, 2, \ldots K)$ to be minimized. Note, for example, that $A\bar{B}$ in F_i and $A\bar{B}$ in F_j are different prime implicants.

2. Determine the elements of the sets $\{P_j\}$ which consist of all product terms in which P_j is contained including P_j itself. These product terms consist of P_j "ANDED" with all combinations of the variables not in P_j, taken one at a time, two at a time, etc. Subscript each element of $\{P_j\}$ with j. For example, for three variables, if $P_1 = A$, then

$$\{P_1\} = \{A\} = (A_1,\ AB_1,\ A\overline{B}_1,\ AC_1,\ A\overline{C}_1,\ ABC_1,\ A\overline{B}C_1,\ AB\overline{C}_1,\ A\overline{B}\,\overline{C}_1)$$

3. Determine a new set of product terms $\{P\}$ which consists partly of the union of all the non-zero elements generated by the intersection of the $\{P_j\}$'s, $(j = 1, 2, \ldots m)$, where combinations of the $\{P_j\}$'s are taken, first two at a time, then three at a time, . . . etc. Each element in $\{P\}$ is subscripted with the sequence of j's determined from the $\{P_j\}$'s used in the intersection from which it was derived. Only take one permutation for each set of indices. Also, each element in $\{P\}$ may contain at most just one subscript associated with each function.

This restriction is included since common factors reduce costs, and have meaning, only when used in two or more different functions. The set of product terms derived by this intersection process is all the possible common factors of the functions. Append to the beginning of this set of common factors, the original set of prime implicants. This union of sets is $\{P\}$.

4. Assign to each element in $\{P\}$ a name $B_r^{\alpha, \beta, \ldots, \rho}$, $(r = 1, 2, \ldots R)$, where the superscripts are the subscripts of the corresponding product terms. Assign to each $B_r^{\alpha, \beta, \ldots, \rho}$ a bivalved variable $B'_r \in (0,1)$, where

$B'_r = 1 \Longrightarrow B_r^{\alpha, \beta, \ldots, \rho}$ is in the minimal form of the functions

$B'_r = 0 \Longrightarrow B_r^{\alpha, \beta, \ldots, \rho}$ is not in the minimal form

Let $a_{v_r} = 1$ if $B_r^{\alpha, \beta, \ldots, \rho}$ contains vertex v

$a_{v_r} = 0$ if $B_r^{\alpha, \beta, \ldots, \rho}$ does not contain vertex v

For $B_r^{\alpha, \beta, \ldots, \rho}$ to contain vertex v, one of the superscripts must be the same as one of the subscripts of the original prime implicants P_j which contained the vertex. This is true since it is necessary to differentiate between vertices which have the same coordinate values but which are in different functions.

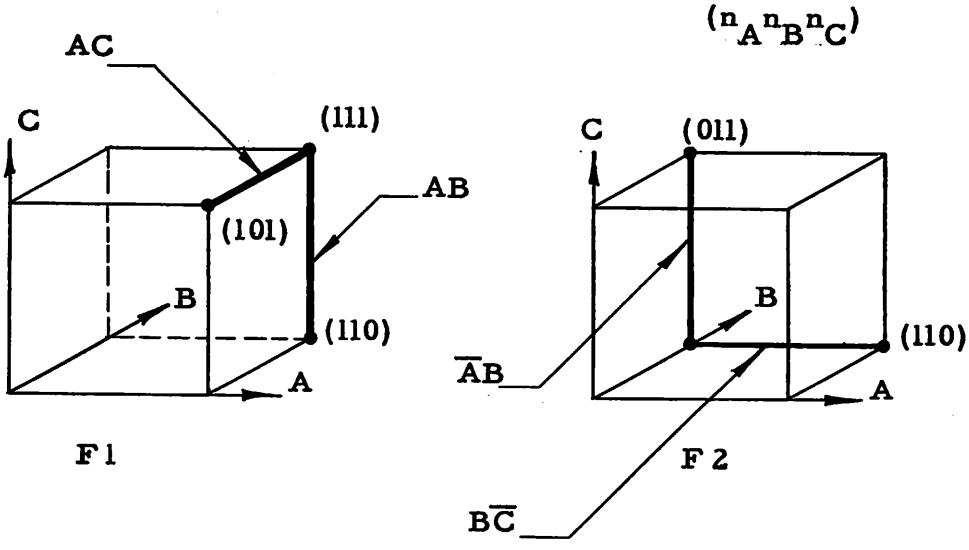
5. Construct the constraint inequalities

$$\sum_{r=1}^{R} a_{v_r} B'_r \geq 1 \qquad \text{for all } v \text{ in } F_k, \text{ and for all } k.$$

6. Construct the objective function. Special care must be given to the cost associated with common factors, since they are used in more than one "OR" gate. The number of "OR" gates is equal to the number of times a product is used as a common factor, and is equal to the number of superscripts in $B_r^{\alpha, \beta, \ldots, \rho}$.

Example 5.

Consider the two functions shown below.



Step 1) $F_1 = \underbrace{AB}_{P_1} + \underbrace{AC}_{P_2}$ $\qquad F_2 = \underbrace{B\overline{C}}_{P_3} + \underbrace{\overline{A}B}_{P_4}$

Step 2) $\{P_1\} = (AB_1,\ ABC_1,\ AB\overline{C}_1)$

$\{P_2\} = (AC_2,\ ABC_2,\ A\overline{B}C_2)$

$\{P_3\} = (B\overline{C}_3,\ AB\overline{C}_3,\ \overline{A}B\overline{C}_3)$

$\{P_4\} = (\overline{A}B_4,\ \overline{A}BC_4,\ \overline{A}B\overline{C}_4)$

Step 3) $\{P_1\} \cap \{P_3\} = AB\overline{C}_{1,3}$

$\{P_1\} \cap \{P_4\} = 0$

$\{P_2\} \cap \{P_3\} = 0$

$\{P_2\} \cap \{P_4\} = 0$

Steps 3) and 4)

$$\{P\} = (\underbrace{AB_1}_{B_1^1},\ \underbrace{AC_2}_{B_2^2},\ \underbrace{B\overline{C}_3}_{B_3^3},\ \underbrace{\overline{A}B_4}_{B_4^4},\ \underbrace{AB\overline{C}_{1,3}}_{B_5^{1,3}})$$

Step 5)

							Vertex	
B_1'					$+B_5'$	≥ 1	(110)	F_1
B_1'	+	B_2'				≥ 1	(111)	
		B_2'				≥ 1	(101)	
			B_3'		$+B_5'$	≥ 1	(110)	F_2
			B_3'	+ B_4'		≥ 1	(010)	
				B_4'		≥ 1	(011)	

Step 6) Let the cost of each term in an "AND" and "OR" gate be one. Then the objective function is

$$3B_1' + 3B_2' + 3B_3' + 3B_4' + 5B_5' = Z\ (\text{MIN.})$$

Note the cost associated with $B_5^{1,3} = AB\overline{C}_{1,3}$ is three for the "AND" gate, plus two for the "OR" gates.

The solution is

$$B_2' = B_4' = B_5' = 1,\ B_1' = B_3' = 0,\ \min\ Z = 11$$

$F_1 = AC + AB\overline{C}$, $F_2 = \overline{A}B + AB\overline{C}$; where $AB\overline{C}$ is a common factor.

Summary:

When simplifying more than one function at a time, a reduction in the cost of implementation is realized if common factors are used. Also, minimal forms need not consist entirely of prime implicants. An algorithm is presented for determining all possible combinations of common factors which exist among a set of Boolean functions. The L. P. for finding the optimal solution is then given.

VI. DUALITY

In this section, the dual of a linear program is introduced, and six pertinent theorems are presented. Conditions are given for simplifying the minimization L. P. prior to solution. From this material, additional insight is obtained in understanding the linear programming minimization formulation.

To each linear program (L. P.), called the primal, there is associated a unique dual L. P. Since the dual of the dual is the primal, either L. P. may be considered the primal. Let the primal be

$$P_j' \geq 0 \qquad j = 1, 2, \ldots, n$$

$$V'_1: \quad a_{11}P'_1 + a_{12}P'_2 + \ldots + a_{1n}P'_n \geq b_1 \quad \text{(vertex 1)}$$

$$V'_2: \quad a_{21}P'_1 + a_{22}P'_2 + \ldots + a_{2n}P'_n \geq b_2 \quad \text{(vertex 2)}$$

$$\vdots \qquad \vdots \qquad\qquad\qquad \vdots$$

$$V'_m: \quad a_{m1}P'_1 + a_{m2}P'_2 + \ldots + a_{mn}P'_n \geq b_m \quad \text{(vertex m)}$$

$$C_1P'_1 + C_2P'_2 + \ldots + C_nP'_n = Z \text{ (min.)}$$

The dual L. P. is

$$V'_i \geq 0 \qquad i = 1, 2, \ldots, m$$

$$a_{11}V'_1 + a_{21}V'_2 + \ldots + a_{m1}V'_m \leq C_1$$

$$a_{12}V'_1 + a_{22}V'_2 + \ldots + a_{m2}V'_m \leq C_2$$

$$\vdots \qquad\qquad\qquad\qquad \vdots$$

$$a_{1n}V'_1 + a_{2n}V'_2 + \ldots + a_{mn}V'_m \leq C_n$$

$$b_1V'_1 + b_2V'_2 + \ldots + b_mV'_m = W \text{ (max.)}$$

From the general theory related to the primal-dual linear programs, the following two important theorems [3] may be derived:

Theorem 1: If an optimal solution exists for the $\left\{\begin{matrix}\text{primal}\\\text{dual}\end{matrix}\right\}$, then one exists for the $\left\{\begin{matrix}\text{dual}\\\text{primal}\end{matrix}\right\}$, and min. Z = max. W.

Theorem 2: If a slack variable which has been added to the i^{th} constraint of the $\left\{\begin{matrix}\text{primal}\\\text{dual}\end{matrix}\right\}$ is different from zero in any optimal solution of the $\left\{\begin{matrix}\text{primal}\\\text{dual}\end{matrix}\right\}$, then in the $\left\{\begin{matrix}\text{dual}\\\text{primal}\end{matrix}\right\}$ the i^{th} variable is zero in every optimal solution.

By analyzing the dual L. P., a few interesting characteristics of the minimization process are brought to light. Note that by Theorem 1, it is possible to solve the dual L. P. and to determine min. Z. It is also possible to determine the P_j''s if all of the V_i''s are known [4].

The following theorem will prove to be useful in simplifying linear programs.

Theorem 3: If the k^{th} inequality in the primal is redundant, then $V_k' = 0$ in at least one of the optimal solutions of the dual.

Definition: An inequality is redundant with respect to a set of inequalities if it is implied by the remaining inequalities in the set. There are two categories into which redundant inequalities usually fall.

1. Weaker condition: If the i^{th} inequality is implied by the j^{th}, then the i^{th} inequality is redundant. Example:

Consider the set of inequalities

$\{P_1' \geq 0,\ P_2' \geq 0,\ P_1' + P_2' \leq 7,\ P_1' \leq 8\}$. Since $P_1' \leq 8$ is implied by the remaining three inequalities, it is redundant.

2. Positive linear combination: If the i^{th} inequality can be expressed as a positive linear combination of a set of inequalities, then it is redundant. Let the k^{th} inequality be (V_k'). Then if

$$(V_i') = \sum_{\substack{j=1 \\ j \neq i}}^{m} a_j (V_j'),\ a_j > 0 \quad \text{for all } j,$$

(V_i') is redundant to the m inequalities (V_j').

Example: (V_1') $P_1' \geq 0,\ P_2' \geq 0$

(V_1') $P_1' + P_2' \leq 5$

(V_2') $P_2' \leq 3$

(V_3') $P_1' + 2P_2' \leq 8$

Since $(V_3') = (V_1') + (V_2')$, (V_3') is redundant. Note that neither (V_1') nor V_2' are redundant since

$$\left\{\begin{array}{l} P_1' = 6 \\ \\ P_2' = 1 \end{array}\right\}$$

satisfies (V_2') and (V_3'), but not (V_1'), and $\left\{\begin{array}{l} P_1' = 0 \\ \\ P_2' = 4 \end{array}\right\}$

satisfies (V_1') and (V_3'), but not (V_2').

The reason for $a_j > 0$ is that if an inequality is multiplied by minus one, the direction of the inequality is reversed as well as the sign of all terms.

Proof: Since the k^{th} inequality is redundant, it may be deleted from the L. P. without effecting the value of min. Z, and hence V_k' will not exist, i. e. $V_k' = 0$. Therefore, when no inequalities are deleted from the primal, since max. W = min. Z, there is at least one optimal solution to the dual with $V_k' = 0$.

Theorem 4:

Let $a_{ij} \geq 0$ all i, j

$c_j > 0$ all j

$b_j > 0$ for at least one i.

Case 1. If $b_k < 0$, $V_k' = 0$ in all optimal solutions of the dual.

Case 2. If $b_k = 0$, and $a_{ij} > 0$ for all k, j, then $V_k' = 0$ in all optimal solutions of the dual.

Case 3. Rule.

If $b_k = 0$, and $a_{ij} \geq 0$ for all i, j, then $V_k' = 0$ is a permissible value in all optimal solutions of the dual, though not always a necessary condition. The rule is that we will always put $V_k' = 0$ for this condition.

: Proof:

Case 1. Assume that there is an optimal solution to the dual with $V_k' > 0$. Since $b_k < 0$, by decreasing V_k' to 0 no constraint inequalities are violated, and W(Max.) is increased. Hence, the original solution could not have been optimal.

Case 2. If $b_k = 0$, then V_k' does not appear in the objective function W(Max.). Assume that there is an optimal solution to the dual with $V_k' > 0$. At least one of the constraints must be an equality. Let this be the p^{th} constraint. (If more than one constraint is at an equality, pick the one which increases the most for an equal and constant increase and decrease in V_i' and V_k' respectively.) Since $b_i > 0$, and $a_{kp} > 0$, decrease V_k' and increase V_i' keeping the p^{th} constraint an equality. By this process W is increased. Hence the original solution was not optimal.

Continue this process until either

a) $V_k' = 0$ and hence the process is finished.

b) The t^{th} constraint becomes an equality.

Repeat procedure on the t^{th} constraint. (Note that for the same increments in V_k' and V_i', the left hand side of the t^{th} constraint changes a greater amount than the left hand side of the p^{th} constraint. Hence, the p^{th} constraint will again become an inequality.) The process ends when $V_k' = 0$.

Case 3. Since $A_{ij} \geq 0$, if $a_{kp} = 0$, V_k' may be increased until a second constraint becomes an equality. As V_k' is increased, W remains at its optimal maximum value. If there is an optimal solution with $V_k'(\max.) > 0$, then there are an infinite number of solutions, corresponding to $0 \leq V_k' \leq V_{k(\max)}'$. The solution(s) for $V_k' = 0$ will be chosen.

Example 6a, b:

(a)

Primal

$P'_j \geq 0, \quad j = 1, 2, 3$

$V'_1: \; P'_1 + P'_2 \geq 1$

$V_2: \; P'_2 + P'_3 \geq 1$

$V'_3: \; P'_1 + P'_2 + P'_3 \geq 0$

$P'_1 + P'_2 + P'_3 = Z(\text{Min.})$

Optimal Solution:

$P'_2 = 1, \; P'_1 = P'_3 = 0, \; \text{min. } Z = 1$

Dual

$V'_i \geq 0, \quad i = 1, 2, 3$

$V'_1 + V'_3 \leq 1$

$V'_1 + V'_2 + V'_3 \leq 1$

$V'_2 + V'_3 \leq 1$

$V'_1 + V'_2 = W(\text{Max.})$

Optimal Solutions:

$V'_3 = 0$

All non-negative V'_1, V'_2

satisfying $V'_1 + V'_2 = 1.$

max. $W = 1$

(b)

Primal

$P'_j \geq 0, \quad j = 1, 2, 3$

$V'_1: \; P'_1 + P'_2 \geq 1$

$V'_2: \; P'_2 \geq 1$

$V'_3: \; P'_1 + P'_3 \geq 0$

$P'_1 + P'_2 + P'_3 = Z \; (\text{Min.})$

Optimal Solution:

$P'_2 = 1, \; P'_1 + P'_3 = 0, \; \text{min. } Z = 1$

Dual

$V'_i \geq 0, \quad i = 1, 2, 3$

$V'_1 + V'_3 \leq 1$

$V'_1 + V'_2 \leq 1$

$V'_3 \leq 1$

$V'_1 + V'_2 = W \; (\text{Max.})$

Optimal Solutions:

All non-negative V'_1, V'_2

satisfying $V'_1 + V'_2 = 1,$

and $0 \leq V'_3 \leq 1 - V'_1$

max. $W = 1$

For Example 6a, since (V'_3) has a non zero slack variable, from Theorem 2 we have $V'_3 = 0$ in the optimal

solution of the dual. This is not the case for Example 6b, where V_3' may take on an infinite number of values between zero and $(1-V_1')$, for V_1' not equal to one. For the solution $V_1' = V_2' = 1/2$, $V_3' = 0$, Theorem 2 gives $P_1' = P_3' = 0$ for both cases. Note finally that max. W = min. Z as stated by Theorem 1. Since (V_3') for (a) and (b) are redundant inequalities, by Theorem 4 (case 3), $V_3' = 0$.

If the primal L. P. is associated with the L. P. derived from the Boolean logic minimization problem, then

$$a_{ij} = 0 \text{ or } 1, \qquad (a_{ij} \geq 0)$$

$$b_i = 1$$

and $C_j > 0$ (are normalized cost coefficients in dollars) for all i, j.

For each prime implicant P_j there is an associated variable P_j', and for each vertex in the Boolean function, there is a constraint inequality. It is possible to associate with each vertex a variable V_i', which has the units of dollars per unit. (Note that $-V_i'$ corresponds to the standard simplex multipliers.) Let (V_i') be the name of the i^{th} vertex of the function being minimized.

Just as there are essential, non-essential irredundant, and non-essential redundant prime implicants, analogous definitions can be defined to vertices.

A vertex is

a. Essential if it is contained in only one prime implicant.

b. Non-essential if it is contained in two or more prime implicants.

1) Irredundant if all of the prime implicants containing this vertex are non-essential.

2) Redundant if at least one of the prime implicants containing this vertex is essential.

Note that the definition of an irredundant prime implicant is analogous to the definition of a redundant vertex, and vice versa.

Theorem 5: All (V_i') corresponding to redundant vertices may be dropped from the primal.

Proof:

Let (V_i') be $P'_{i_1} + P'_{i_2} + \ldots + P'_{i_g} + \ldots + P'_{i_t} \geq 1$

where

$P_{i_1}, P_{i_2}, \ldots P_{i_g}$, $(1 \leq g \leq t)$ are essential prime implicants. Therefore $P'_{i_1} = P'_{i_2} = \ldots = P'_{i_g} = 1$

Hence $P'_{i_{g+1}} + \ldots + P'_{i_t} \geq 1 - g \leq 0$ (2)

But $P'_j \geq 0$ for all j implies (3)

$$P'_{i_{g+1}} + \ldots + P'_{i_t} \geq 0$$

Note that (2) is an equivalent or weaker condition than (3), and hence may be dropped from the primal.

Corollary 1: In the optimal solution to the dual, the V'_i associated with a redundant cell is zero. If (V'_i) is left in the primal, then

$$V'_i : P'_{i_{g+1}} + \ldots + P'_{i_t} \geq 1 - g, \; g \geq 0$$

By Theorem 4, $V'_i = 0$ for $g > 0$, and by rule 1 for case 3, $V'_i = 0$ for $g \neq 0$.

It can be concluded that constraint inequalities (V'_i) need be written for only those vertices V'_i which are not redundant.

Example:

Consider the function

$$F = \underbrace{AB}_{P_1} + \underbrace{B\overline{C}}_{P_2} + \underbrace{\overline{A}\overline{B}}_{P_3} + \underbrace{\overline{A}\overline{C}}_{P_4}$$

Let the per unit cost associated with literal $B(_{cg})$ be two, and all other c's be one. The primal L. P. is

$$P'_j \geq 0 \quad j = 1, 2, 3, 4$$

V'_1 : $P'_1 \geq 1$ vertex (111)

V'_2 : $P'_1 + P'_2 \geq 1$

V'_3 : $P'_2 + P'_4 \geq 1$

V'_4 : $P'_3 + P'_4 \geq 1$

V'_5 : $P'_3 \geq 1$ vertex (001)

$$4P'_1 + 4P'_2 + 3P'_3 + 3P'_4 = Z \text{ (Min.)}$$

The optimal solution is

$$P'_1 = P'_3 = P'_4 = 1, \; P'_2 = 0, \text{ min. } Z = 10$$

The dual L. P. is

$$V'_i \geq 0 \qquad \leq 4$$

$$V'_1 + V'_2 \qquad \leq 4$$

$$V'_2 + V'_3 \qquad \leq 4$$

$$V'_4 + V'_5 \leq 3$$

$$V'_3 + V'_4 \leq 3$$

$$V'_1 + V'_2 + V'_3 + V'_4 + V'_5 = W \text{(Max.)}$$

The optimal solution is

$$V'_1 = 4, \; V'_3 = V'_5 = 3, \; V'_2 = V'_4 = 0, \text{ max. } W = 10$$

The function in three space is shown in Figure 5, and the optimal solutions are indicated.

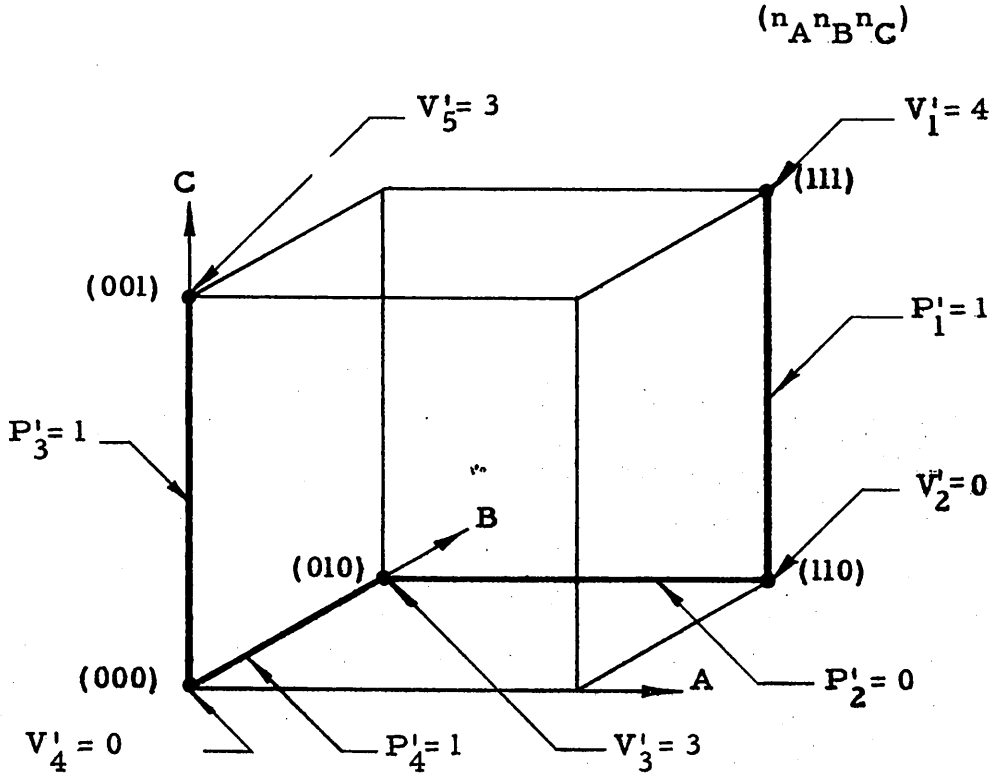


Figure 5. Three space representation of minimal solution to Boolean function F.

From (V_1'), $P_1' \geq 1 \rightarrow P_1' = 1$.
Substituting into (V_2'),

$$P_2' \geq 0$$

which is a redundant inequality. Hence, by Corollary 1, $V_2' = 0$ in the optimal solution to the dual.

Analogously, $P_3' = 1$, and hence $V_4' = 0$. Note that this checks with the solutions obtained. Finally, from Theorem 5, (V_2') and (V_4') could have been deleted from the primal, without affecting the solution.

The solution to the dual may be interpreted as implying that an associated shadow price or imputed value of 4, 0, 3, 0, 3 with vertices V_1', V_2', V_3', V_4', V_5', respectively. Hence, the redundant vertices V_2' and V_4' are gotten for nothing.

Linear programming is a lumping rather than a smoothing operation. This can clearly be seen from the solution to the dual, since the shadow prices are not distributed over all vertices.

Since $P_3' = P_4' = 1$ in the optimal solution, the slack variable in (V_4') is non-zero, and hence from Theorem 2, $V_4' = 0$. This again checks with the results. Also, since $V_2' + V_3' = 3$, then $P_2' = 0$, again by Theorem 2.

The fact that $V_i' = 0$ does not imply that (V_i') is redundant. As an example, consider the function shown in Figure 6.

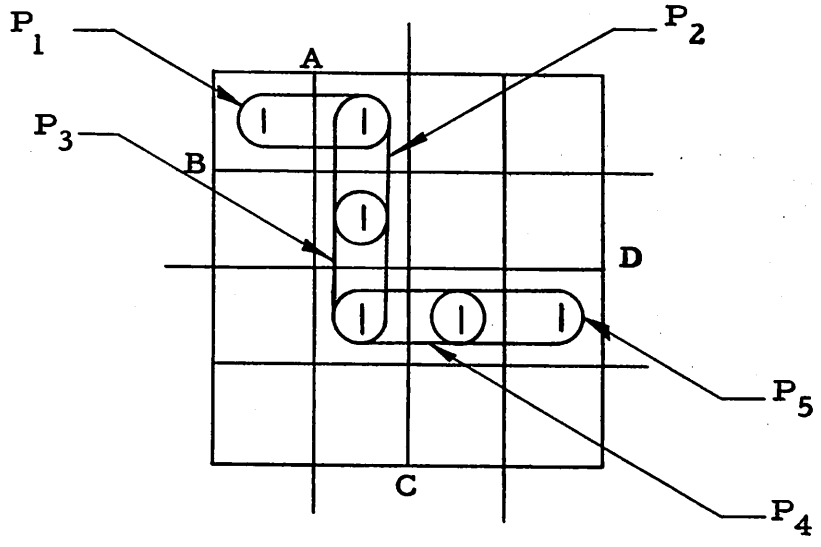


Figure 6. Karnaugh diagram representation of Boolean function F.

$$F = \underbrace{AB\bar{D}}_{P_1} + \underbrace{ABC}_{P_2} + \underbrace{ACD}_{P_3} + \underbrace{\bar{B}CD}_{P_4} + \underbrace{\bar{A}\bar{B}D}_{P_5}$$

The constraint inequalities for the non-redundant vertices are

$$\begin{aligned} P_1' &\geq 1 \\ P_2' + P_3' &\geq 1 \\ P_3' + P_4' &\geq 1 \\ P_5' &\geq 1 \end{aligned}$$

Let $4P_1' + 4P_2' + 4P_3' + 4P_4' + 4P_5' = Z(\text{Min.})$

Since $P_1' = P_5' = 1$, the modified primal is

V_1' :
$$\begin{aligned} P_2' + P_3' &\geq 1 \\ P_3' + P_4' &\geq 1 \end{aligned}$$

$$4P_2' + 4P_3' + 4P_4' = Z'(\text{min.})$$

The optimal solution is

$$P_3' = 1, \quad P_2' = P_4' = 0, \quad \text{min. } Z' = 4$$

The dual L. P. is

$$V_i' \geq 0 \quad i = 1, 2$$

$$\begin{aligned} V_1' &\leq 4 \\ V_1' + V_2' &\leq 4 \\ V_2' &\leq 4 \end{aligned}$$

$$V_1' + V_2' = W' \text{ (Max.)}$$

There are an infinite number of optimal solutions to the dual, each satisfying the relationships,

$$V_1' + V_2' = 4$$

$$0 \leq V_1' \leq 4$$

Two particular solutions are

$$V_1' = 0, \quad V_2' = 4$$

and $V_1' = 4, \quad V_2' = 0$

Note that $V_i' = 0$ $(i = 1, 2)$ does not imply that (V_i') $(i = 1$ or $2)$ is redundant, nor that vertex V_i' is redundant; however, from Theorem 2, $P_2' = P_4' = 0$ in all optimal solutions to the primal. The question now arises as to what value should be assigned to V_1' and V_2' in order to be consistent with the definition of associating V_i' with the shadow price of vertex V_i'.

Since the cost of all literals was assumed to be constant, let

$$V_1' = V_2' = 2$$

Note that the prime implicant $P_3 = ACD$ contains vertices V_1' and V_2', i.e.

$$ACD = \underbrace{ABCD}_{V_1'} + \underbrace{A\overline{B}CD}_{V_2'}$$

If, however, the per unit literal costs are allowed to be

$$c_B = 2, \quad c_{A, \overline{B}, C, D} = 1$$

then the V''s can be determined as follows:

Let $C(\gamma)$ be the cost of implementing the Boolean product term γ, including one diode for an "OR" gate. Then

$$C(ACD) = 4 = V_1' + V_2'$$

$$C(ABCD) = 6$$

$$C(A\overline{B}CD) = 5$$

and hence,

$$V_1' = 4\left(\frac{6}{6 + 5}\right) = \frac{24}{11}$$

$$V_2' = 4\left(\frac{5}{6 + 5}\right) = \frac{20}{11}$$

Summary:

By introducing the dual L. P., it is possible to associate a shadow price or imputed value with each vertex contained in F_k. Essentially, these imputed values are a measure of the significance of each vertex in establishing which prime implicants are to be in the minimal form. A redundant vertex, for example, has an imputed value of zero, and has no effect in determining the necessary constraints of the L. P. Finally, the objective function of the dual is to be maximized. This objective function is analogous to the total activity level usually considered in economic problems. By maximizing the activity level, we are essentially seeking the most efficient covering of the vertices in F_k.

If the dual is easier to solve than the primal, (the $\left\{ \begin{matrix} \text{primal} \\ \text{dual} \end{matrix} \right\}$ has $\left\{ \begin{matrix} n \\ m \end{matrix} \right\}$ variables and $\left\{ \begin{matrix} m \\ n \end{matrix} \right\}$ inequalities), then the V_i''s may be solved for, from which the P_j''s can be determined. The final form of the A_{ij} matrix is also required. From the knowledge of the slack variables in one system, the second system may be simplified. Hence, the dual L. P. and the associated theorems presented are an aid in understanding and simplifying the linear programming minimization procedure.

VII. FAN-IN CONSIDERATIONS

The fan-in restrictions on the "OR" gate required to form F_k may be taken into account in the simplification computation by adding the additional constraint inequalities of the form

$$\sum_j B_j' \leq F_{IN} \quad \text{for each } F_k, \quad \text{where } j \text{ is summed over}$$

all B_j' contained in F_k, and the constant F_{IN} is the fan-in

factor which equals the maximum number of inputs to an "OR" gate. B_j is a generalized product term, consisting of either the output of a flip-flop or "AND" gate.

The choice of F_{IN} is usually based on a compromise between circuit restrictions, costs, and certain requirements demanded by the logical equations. Rather than leave the choice of F_{IN} to human judgment, it is possible to have the L. P. determine the optimal choice for F_{IN}. As pointed out earlier, the L. P. can determine the optimal choice of common factors, and of amplifiers needed for the fan-out problem. It is therefore seen that the minimization technique takes into consideration not only the Boolean functions, but also the circuit costs required in implementing the functions. This appears to be a logical approach to the problem.

The motivation for making F_{IN} a variable in the L. P. will be justified by showing how an additional savings in cost is realized. Assume that all "OR" gates consist of the same standardized package, and hence all have the same upper limit F_{IN} for the maximum number of inputs. Assume, for the present, that F_{IN} is a constant, and solve for the optimal solution to the L. P. Now, F_{IN} must satisfy all of the constraint inequalities for each function F_k, since the solution is feasible. However, the cost of implementing an "OR" gate is a monotonically increasing function of F_{IN}. Hence, F_{IN} can be decreased until at least one of the constraints becomes an equality. Assume that the constraint for function F_k is an equality, while all the other constraints are inequalities, i. e.,

$$\sum_j B_j' = F_{IN} \text{ for } F_k \qquad (4)$$

$$\sum_j B_j' < F_{IN} \text{ for } F_i, \text{ all } i \neq k.$$

Now, if F_{IN} is a variable, it may be possible to find another representation of F_k which, though possibly more expensive to implement, requires fewer "OR" terms in (4), and hence F_{IN} may be reduced. If the total cost saved in reducing F_{IN} (in all "OR" gates) is greater than the increase in cost encountered in implementing F_k, then a better solution has been found. This justifies the purpose of making F_{IN} a variable. Note that the same argument and justification holds if more than one constraint is at an equality.

In order to introduce F_{IN} into the L. P. as a variable, the relationships between gating costs and F_{IN} must be expressed as a set of linear constraints. In general, let the linearized per unit amplifier cost c_a be

$$c_a = f_1 (F_{IN})$$

Also, let the cost associated with each "OR" gate be C_{OR}, and with each "AND" gate be C_{AND}, where

$$C_{OR} = f_2 (F_{IN})$$

$$C_{AND} = f_3 (F_{IN})$$

Note that c_a, C_{OR}, and C_{AND} are functions of many factors, where only F_{IN} is allowed to vary.

A method is now presented for expressing the linearized cost C vs. fan-in relation as a set of linear constraints [5]. Expressing any value of C and F_{IN} as a weighted average of two successive breakpoints, it follows that (see Figure 7)

$$F_{IN} = \lambda_o a_o + \lambda_1 a_1 + \ldots + \lambda_k a_k$$

$$C = \lambda_o b_o + \lambda_1 b_1 + \ldots + \lambda_k b_k$$

$$1 = \lambda_o + \lambda_1 + \ldots \lambda_k$$

$$0 \leq \lambda_1 \leq 1 \quad i = o, 1, \ldots k$$

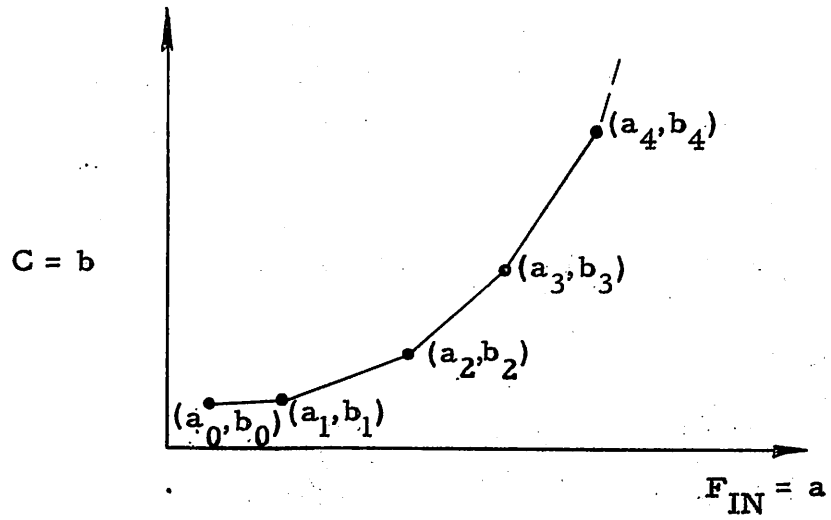


Figure 7. Linearized $C = f(F_{IN})$

In order to ensure the selection of two successive breakpoints it is required that,

$$\lambda_o \leq \delta_o$$
$$\lambda_1 \leq \delta_o + \delta_1$$
$$\lambda_2 \leq \delta_1 + \delta_2$$
$$\vdots$$
$$\lambda_{k-2} \leq \delta_{k-3} + \delta_{k-2}$$
$$\lambda_{k-1} \leq \delta_{k-2} + \delta_{k-1}$$
$$\lambda_k \leq \delta_{k-1}$$

where $\delta_i \in (0, 1)$ and

$$\delta_o + \delta_1 \cdots \delta_{k-1} = 1$$

Since the δ_i's are integers while the λ_j's are fractional, mixed integer programming must be employed.

The objective function is now determined. Since each variable is implemented by the use of a state device, such as a flip-flop, and if it is assumed that the device has both a true and a false output driven by amplifiers, then to the objective function can be added the term $2n\,C_f$, where n is the number of variables, and C_f is the flip-flop amplifier cost.

$$C_f = f_4(F_{IN})$$

C_f need not equal c_a.

The number of "AND" gates is $Y = \sum_{j=1}^{m} B_j'$, and the associated total cost is

$$C_{AND}\,Y$$

The number of "OR" gates is $(Y-W)$, where $W = m_o$ is the number of functions F_k containing only one "OR" term. Assume $m_o = 0$. The associated total cost is

$$C_{OR}\,Y$$

The new objective function is (see Equation (13), of [1])

$$2nC_f + C_{AND}\,Y + C_{OR}\,Y + \sum_i C_{iT} + c_d Y = Z\,(\text{Min.})$$

$$\text{where } C_{iT} = c_d N_i + c_a\,(\delta_{i_1} + \delta_{i_2} + \ldots + \delta_{i_{t_i}})$$

Since c_a, C_{OR}, C_{AND}, Y, and the δ's are variables, the objective function is quadratic, and hence nonlinear. For information on quadratic programming, see [6], [7], and [8].

In order to analyze the objective function in more general terms, the following change of variables is now made.

Let $B_j' \rightarrow x_j$ $\quad j = 1, 2, \ldots, m$

$\delta_{i\,h_i} \rightarrow x_j$ $\quad j = m+1, \ldots J$, and where for each combination of i and h there is a unique j.

$C_{AND} \rightarrow x_a$

$C_{OR} \rightarrow x_o$

$C_f \rightarrow x_f$

$c_a \rightarrow x_c$

The objective function is thus

$$2n\, x_f + x_a (x_1 + x_2 + \ldots x_m) + x_o (x_1 + x_2 + \ldots + x_m)$$

$$+ c_d (x_1 + x_2 + \ldots + x_m) + c_d (a_1 x_1 + a_2 x_2 + \ldots + a_m x_m)$$

$$+ x_c (x_{m+1} + \ldots + x_J) = Z\,(\text{Min.}) \qquad (5)$$

where a_i is the number of literals in x_i.

Let the vector $x = (x_1, x_2 \ldots, x_J, x_f, x_a, x_o, x_c)$

Since every quadratic form can be expressed in terms of a symmetric matrix C, we have,

$$a\,x + x^T C x = Z\,(\text{Min.})$$

where T stands for transpose.

Now, a quadratic form is positive semi-definite if

$$x^T C\, x \geq 0 \text{ for all } x.$$

Referring to the quadratic terms in (5), let

$$x_a = 1,\ x_1 = -1, \text{ all other } x\text{'s} = 0.$$

$$x^T C\, x = -1$$

and hence $x^T C\, x$ is not positive semi-definite. Due to this fact, the standard algorithms used to solve quadratic programming

problems, but which assume the positive semi-definite property to hold, cannot be applied.

The reason for this is that in general, if $x^T Cx$ is not positive semi-definite, the solution to the program may not be the global minimum, but rather a local minimum or possibly a stationary value which is neither a local nor global minimum.

Fortunately, the quadratic function being considered has a very special form. Note that there are no square terms, and that it is highly factorable. Let all $\delta_{i_{t_i}} = 0$, that is, do not consider the addition of amplifiers into the circuits. The quadratic term is then

$$Q(x) = \underbrace{(x_a + x_o)}_{X} \underbrace{(x_1 + x_2 + \ldots + x_m)}_{Y} = X\,Y$$

This function is shown in Figure 8.

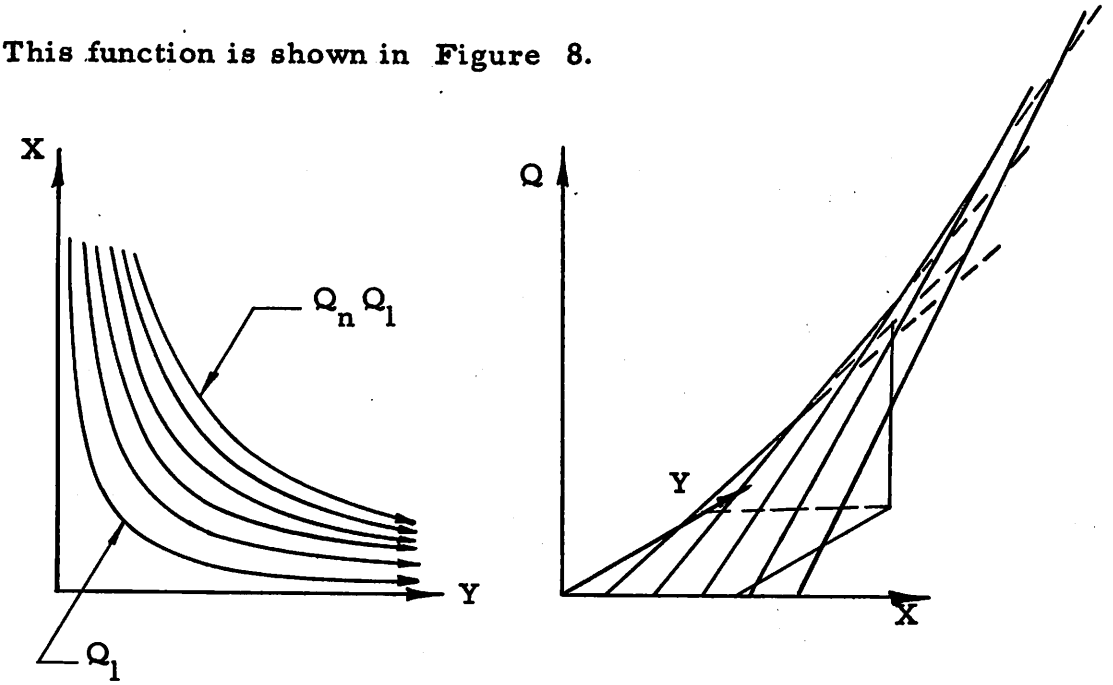


Figure 8. $Q(x) = XY$

Note that the intersection of the surface $Q(x)$ with the planes $Y = \text{const.}$ and $X = \text{const.}$ are straight lines. Also, $XY = \text{const.}$

is a hyperbola. From these properties, it is conjectured that the quadratic program can be solved for the optimal solution.

Beale [8] presents an algorithm for solving quadratic programs when the objective function is convex. A function is convex if

$$f(\lambda x + (1 - \lambda) y) \leq \lambda f(x) + (1 - \lambda) f(y)$$

Now positive semi-definiteness implies convexity, but not vice versa. However, as can be seen from Figure 8, the function

$$Q(x) = XY$$

is not convex, and hence the procedure of Beale does not apply. However, it appears that no local minima exists, and hence a procedure based upon the principle of steepest descent should produce an optimal solution. Unfortunately, the author is not aware of any published algorithm for solving the type of quadratic form under consideration.

Though a procedure for finding the optimal solution will not be presented in this paper, a theorem will be given for testing whether or not a feasible solution is optimal. The test is accomplished by solving a dual linear program derived from a transformation of the quadratic program under consideration.

$$\text{Let } x = \begin{Bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{Bmatrix}, \quad b = \begin{Bmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ \cdot \\ b_m \end{Bmatrix}, \quad a = (a_1, a_2, \ldots, a_n)$$

$$\pi = (\pi_1, \pi_2, \ldots, \pi_m)$$

$$A = (a_{ij}) \qquad i = 1, 2, \ldots, n \quad j = 1, 2, \ldots, m$$

$$C = (c_{ik}) \qquad i, k = 1, 2, \ldots, n$$

where $c_{ik} = c_{ki}$

Theorem 6:

$$x \geq 0 \tag{6a}$$

$$Ax \geq b \tag{6b}$$

$$2ax + x^T Cx = Z \text{ (Min.)} \tag{6c}$$

If x_o solves (6), then there exists

$$\pi_o \geq 0 \text{ such that}$$

$$\pi_o A \leq a + x_o^T C$$

$$\pi_o b = ax_o + x_o^T Cx_o$$

Proof: x_o is the optimal solution, and let x be any non-optimal feasible solution. Then

$$\lambda x + (1 - \lambda) x_o \qquad 0 < \lambda \leq 1$$

is a feasible solution. Substituting into (6c),

$$2a [\lambda x + (1 - \lambda) x_o] + (\lambda x + (1 - x) x_o)^T C (\lambda x + (1 - \lambda) x_o) \geq 2ax_o + x_o^T Cx_o$$

$$2a [\lambda x + (1 - \lambda) x_o] + \lambda^2 x^T Cx + \lambda (1 - \lambda) x_o^T Cx + \lambda (1 - \lambda) x^T Cx_o$$

$$+ (1 - \lambda)^2 x_o^T C x_o \geq 2ax_o + x_o^T Cx_o$$

Now $x_o^T Cx = x^T Cx_o$ since C is symmetric. Cancelling common terms on both sides, then

$$2a [\lambda (x - x_o)] + \lambda^2 x^T Cx + 2\lambda (1 - \lambda) x_o^T Cx + \lambda (\lambda - 2) x_o^T Cx_o \geq 0$$

and cancelling λ,

$$2a[x - x_o] + \lambda\, x^T Cx + 2(1-\lambda)\, x_o^T Cx + (\lambda - 2)\, x_o^T Cx_o \geq 0$$

Let $\lambda \longrightarrow 0$

Then

$$a\,[x - x_o] + x_o^T Cx - x_o^T Cx_o \geq 0$$

$$(a + x_o^T C)\, x \geq ax_o + x_o^T Cx_o = \text{min. } Z$$

The original system can therefore be written as

$$x \geq 0$$

$$Ax \geq b$$

$$(a + x_o^T c)\, x = Z \text{ (Min.)}$$

The dual to this linear system is

$$\pi \geq 0$$

$$\pi A \leq a + x_o^T C$$

$$\pi b = W(\text{Max.})$$

The solution is

$$\pi_o = \text{Max. } W = \text{min. } Z = ax_o + x_o^T Cx_o$$

Summary:

In this section it has been shown that an increased saving in total cost of implementation may be realized if some functions are implemented in a non-optimal form. The objective is to reduce the upper bound F_{IN} on the number of inputs to an "OR" gate. By making the fan-in factor, F_{IN} a variable, the resulting quadratic program becomes an exact mathematical model of the problem. It should be noted that the same objective could be reached by manually changing the cost constants in the objective function, and integrating the linear program for different values of the constant F_{IN}.

REFERENCES

1. Breuer, M. A., "The Minimization of Boolean Functions Containing Unequal and Nonlinear Cost Functions--Part I," Series 60, Issue No. 431, Electronics Research Laboratory, University of California, Berkeley, Jan. 1962.
2. Phister, Montgomery, Jr., Logical Design of Digital Computers, John Wiley & Sons, Inc., New York, 1958, pp. 94-96.
3. Hadley, George, Linear Programming, Addison Wesley, Reading, Mass., 1962, 520 pp.
4. Garvin, Walter W., Introduction to Linear Programming, McGraw-Hill Book Co., Inc., New York, 1960, pp. 50-53, 246-257.
5. Dantzig, George B., "On the significance of solving linear programming problems with some integer variables," Econometrica, Vol. 28, No. 1, pp. 30-44; Jan. 1960.
6. Wolfe, Philip, "The simplex method for quadratic programming," Econometrica, Vol. 27, No. 3, pp. 382-398; June 1959.
7. Kuhn, H. W. and Tucker, A. W., "Nonlinear Programming," Second Berkeley Symposium on Mathematical Statistics and Probability, Univ. of California Press, 1951, pp. 481-492.
8. Beale, E. M. L. "On Quadratic Programming," U. S. Naval Research Logistics Quarterly, Vol. 6, pp. 227-243, 1959.