A RELAY-TREE METHOD FOR SIMPLIFYING BOOLEAN FUNCTIONS

*By Hao-Yung Lo

(Received 9 December 1974)

Abstract—*Generally there are two methods for simplifying Boolean functions; one is the Karnaugh Map Method, and the other, the Quine-Mccluskey Tabular Method.*

The purpose of this article, which is modified from a paper by A. H. Scheinman, is to suggest a Relay-Tree Method for simplifying Boolean Functions. The Relay-Tree Method has three advantages over Scheinman's method:

(1) it is not necessary to begin with the most significant variable (the one on the left); the simplifying procedure may start from any variable in the terms of a canonical form.
(2) it will automatically eliminate redundant prime implicants altogathers, so that there is no need to select the essential implicant from the Prime Implicant Table.
(3) it can also applied to simplify Multi-output networks.

Generally there are two methods for simplifying Boolean functions; one is the Karnaugh Map Method [1], and the other, the Quine- McCluskey Tabular Method[2]. The former will probably be the best one if the function has less than six variables. However, it becomes incovenient when there are more than six variables, and difficult to program. The latter method, since it is iterative technique for comparing two terms of a canonical form that differ in only one variable, can be used for a large number of variables, and is convenient for a computer. But the method is a tedious one, and has several defects; its result includes noneesential prime implicants, and the essential prime implicants must be slected from another table. In this article, which its modified from a paper by A. H. Scheinman [3] in BSTJ. a Relay-Tree Method is suggested for simplifying Boolean functions. This method has at least two different points, comparing with Scheinman's:

(1) it is not necessary to begin with the most significant variable (the one on the left); the simplifying procedure may start from any variable in the terms of a canonical form.
(2) it can automatically eliminate redundant prime implicants. You need not select the essential prime implicants by another prime Implicant Table.

Basic theorem: Any Boolean function $f(x, x', y, \cdots, z)$ can be expanded to

*The author is with National Tsing Hua University, Hsin Chu, Taiwan.

$$f(x, x', y, \cdots, z) = x.f(1, 0, y, \cdots, z) + x'.f(0, 1, y, \cdots, z) \cdots \quad (1)$$

This theorem appears in the most test books. Equation(1) may be written:

$$f\ (x,x',y,\cdots z) = x \cdot \phi_1\ (y,\cdots,z)\ + x' \cdot \phi_2\ (y,\cdots,z)\ = x \cdot\ (\phi_c + \phi_R)\ + x\ (\phi_c + \phi_s). \quad (2)$$

Here ϕ_c denotes the terms common to the residual functions ϕ_1 and ϕ_2; and ϕ_r denotes the terms in $\cdot\ \phi_1$ but not in ϕ_2, while ϕ_s denotes the terms in ϕ_2 but not in ϕ_1. Then we have

$$\phi_c = (1 + x + x').\phi_c \quad (3)$$

and equation (2) becomes:

$$f(x, x', y, \cdots z) = \phi_c + x \cdot (\phi_c + \phi_r) + x'\ (\phi_c + \phi_s) \quad (4)$$

This important equation is exactly the same as the one derived by Scheinman. However, in applying it to the simplification of Boolean functions we shall take a different point of view.

We proceed to illustrate the simplifying procedures by several examples:

Example 1. Consider the following function, with eleven terms:

$$T = ABC'D' + AB'CD' + ABCD' + A'B'C'D + AB'C'D + ABC'D + AB'CD + ABCD + A'B'CD' + A'B'CD + A'BC'D$$

Step (1) Convert the function to binary form

$$T = 1100 + 1010 + 1110 + 0001 + 1001 + 1101 + 1011 + 1111 + 0010 + 0011 + 0101$$

It is to be noted that in Scheinman's paper the function is converted to decimal form with the weight given for each variable.

$$T = \sum(12, 10, 14, 1, 9, 13, 11, 15, 2, 3, 10)$$

Step (2) Let the four terms begining with 0 be put into one group, omitting the 0, let this group be called A', and let its terms designated consecutively by 1,2,3,4. Then let the seven terms begining with 1 be put into one group (omitting the 1), called A, and designated by 5.6,$\cdots$,11. Finally, let the terms (if any) that are common to A and A' be put also into a third group, called X, each of them being designated by the two corresponding numbers from A' and A.

Step (3) In group A' and A write a slash through the number designating any terms that also in X. If all the terms in a group are slashed (i. e. if the entire group is contained in x), then slash the letter denoting the group (*e, g.* in Fig 1, the A' is slashed) and eliminate this group further consideration.

Remark 1. In any group the number of simplified prime implicants will not be greater than the number of nuslashed terms; i.e.,

let P=total number of terms in a group, say A', and let Q=number of slashed terms. Then if R represents the number of prime implicants in A', we will have:

$R \leq P-Q$

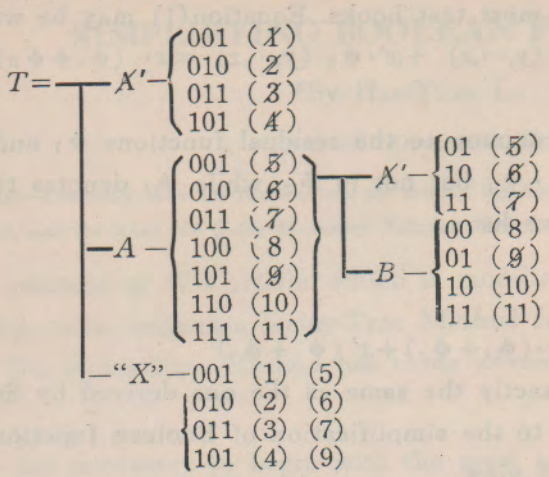


Fig. 1. Expansion of the variables A and B.

Step (4) Thus in Fig. 1, since A contains the unslashed terms, we put the three-terms in A that begin with 0 into a group B' (after omitting the 0) and the four terms in A that begin with 1 into the group B (after omitting the1). Now treat the remaining group in the first line in Fig. 1 namely the terms in $(n-1)$ variables in the same way as the n-variable terms the original function T, (in our example $n=4$).

Remark 2, lf, as in Fig, 1, the group B contains all the 2^n possible arrangement of n zeros and ones, or if B' or B is empty, then there is no need to construct the group X. Otherwise X must be constructed in each case.

Step (5) Deal in the same way with the groups X that appear during the process, and carry out the process until one-variable terms are reached; i.e. $(n-1)$ times:

Step (6) Starting from each of the unslashed groups in the first line-terms in $(n-1)$ variables,trace a path as far as possible through an unslashed $(n-2)$-variable group, an unslashed $(n-3)$-variable group etc. Finally, omit all X-group. Thus in Fig. 1, starting from A we obtain $ABX=AB$, and starting from X we obtain $XB'C=B'C$ and $XX-C'D=C'D$. Then the complete solution is obtained as the sum of these prime implicants (Fig.2).

$$T=AB+B'C+C'D$$

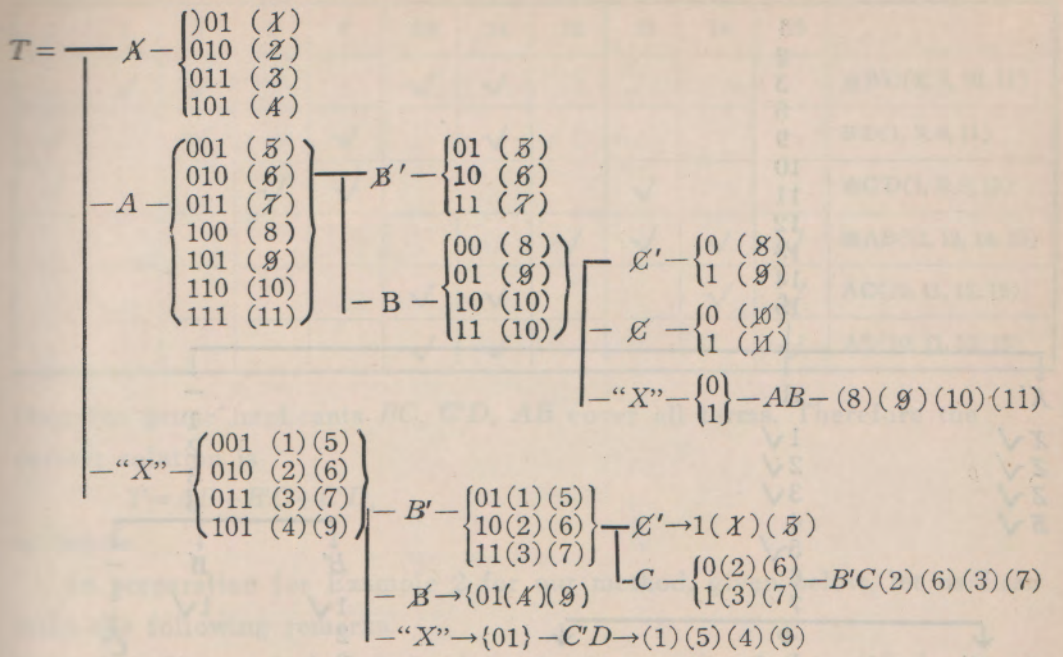


Fig. 2. The comlete expansion of the function T.

If this example is solved by the Tabular Method or by Scheinman's Method, each of them will involve six prime implicants. So three out six implicants must be selected as essential implicants from another table, As a contrast to our method we now consider the above example by Scheinman's method.

(A) Convert the same function to decimal form and arrange the weights numbers in order of increasing magnitude

$T = \sum(1, 2, 3, 5, 9, 10, 11, 12, 13, 14, 15)$

(B) Expand the function from the most significant variable (A in this case); and then from B, $C\cdots$, (See Fig. 3).

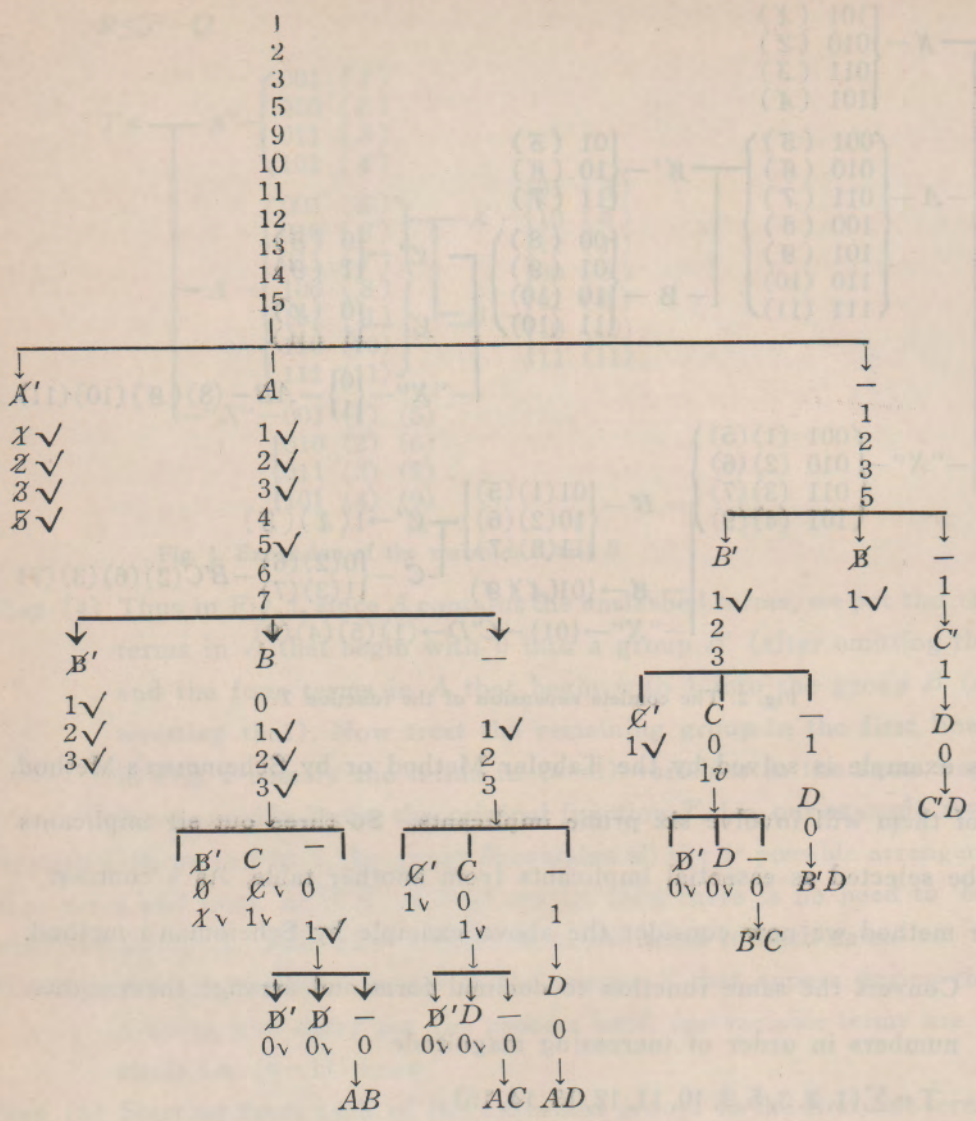


Fig. 3. The complete expansion for Scheinman's Method.

The expansion procedures are similar to ours. Note that the Col. A' simply lists the weights that are smaller than the weight of MSV (the most Significant variable, 8 in this case) and that the numbers in Col. A are obtained by subtracting 8 from the weights that are not less than 8. The numbers in the third column are those that are common to the first two columns.

From the Fig. 3, we get all the prime implicants of the function:

$$T=AB+AC+AD+B'C+B'D+C'D$$

Here is necessary to select essential prime implicants from the following table.

Table 1. Selection of essential prime implicants

1	2	3	5	9	10	11	12	13	14	15	
	√	√			√	√					※B'C(2, 3, 10, 11)
√		√		√		√					B'D(1, 3, 9, 11)
√			√	√				√			※C'D(1, 5: 9, 13)
							√	√	√	√	※AB(12, 13, 14, 15)
					√	√			√	√	AC(10, 11, 13, 15)
					√	√				√	AD(10, 11, 13, 15)

Only the prime implicants BC, $C'D$, AB cover all terms. Therefore the correct solution is

$$T=AB+B'C+C'D,$$

as before

In preparation for Example 2 for our method, given below, let us first make the following remarks.

Remark 3. If any slashed decimal number appears in the simplified primed implicants in grouy A' or group A, then the same number should also be slashed in order to avoid redunant terms. (Fig. 4)

Remark 4. Generally, the third group should always be formed in such a way as to obtain the simplest results, even though one of the two subgroups is eliminated. This is because a single term in the third group represents at least two terms of the original function.

Note that this rule is consistent with Remark 2 above, since if we form the third subgroup of A, headed by X, as in Fig, 4, the implicant term AB causes the numbers 10, 11 to be slashed and the sub-group X is eliminated:

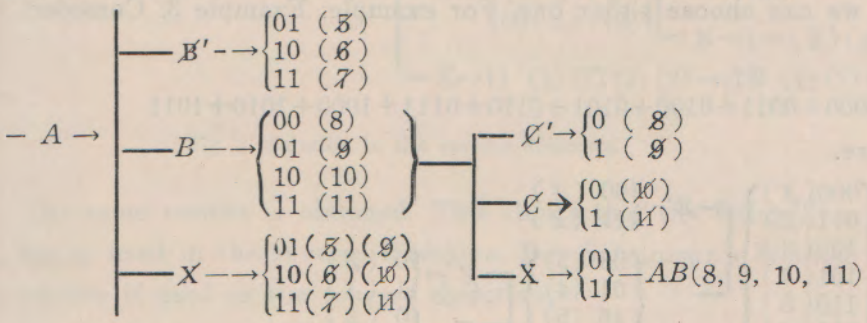


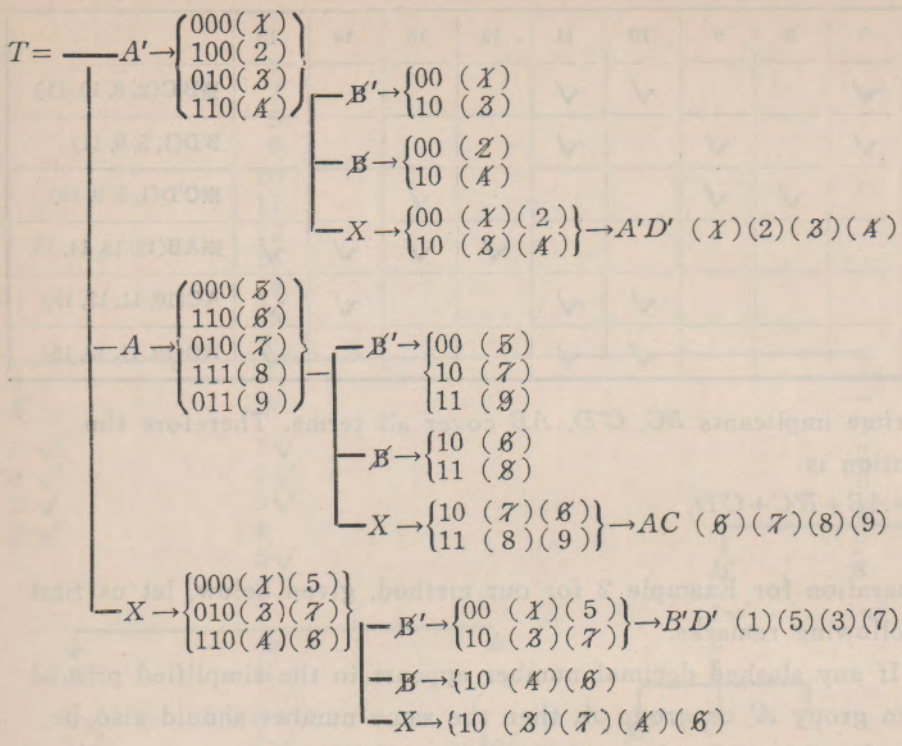
Fig. 4. A Partial Relay-Tree for Remark 4.

To summarize, we always want to form the third subgroup "X", even if that subgroup is eliminated, except when another Subgrouy includes all terms of all residual combinations (See Fig. 5; "X" subgroup of "A'").

As a second example for our method, let us simplify the function

$$T=0000+0100+1000+0010+010+1110+1010+1111+1011$$

for which the figure will be:



Fg. 5. Solution for Example 2

Since the slashed numbers 1, 3, 4, 6, 7, appear in the prime implicants $A'D'$ and AC in group A or A', we should also slash the same numbers appearing in the third group X. Otherwise rednudant prime implicants may exist. From Fig. 5. We see that the solution is

$$T=A'D'+AC+B'D$$

Remark 5. Sometimes the possibility of two solutions may occur. Under this circumstance we can choose either one. For example: Example 3. Consider the function

$$T=0000+0011+0100+0101+0110+0111+1000+1010+1011$$

with the figure:

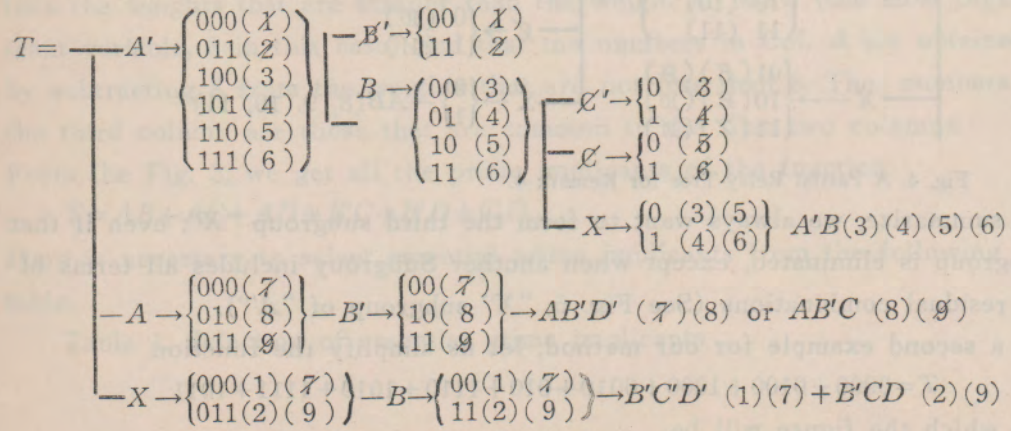


Fig. 6. Solution for Example 3.

The term in the AB' Subgroup may be combined with either the slashed term $\not{7}$ to obtain $AB'D'$ or with the slashed $\not{9}$ to obtain $AB'C$. Both of the solutions are correct:

$$T_1 = A'B + AB'D' + B'C'D' + B'CD$$

or

$$T_2 = A'B + AB'C + B'C'D' + B'CD.$$

We have stated at the begining that simplifying function can be processed in the reverse direction or even starting from any intermediate variable. Let us illustrate the fact by looking Example 1 in the reverse direction.

The same function as in example 1 is rewritten here:

$$T = 1100 + 1010 + 1110 + 0001 + 1001 + 1101 + 1011 + 1111 + 0010 + 0011 + 0101$$

Now expanding:

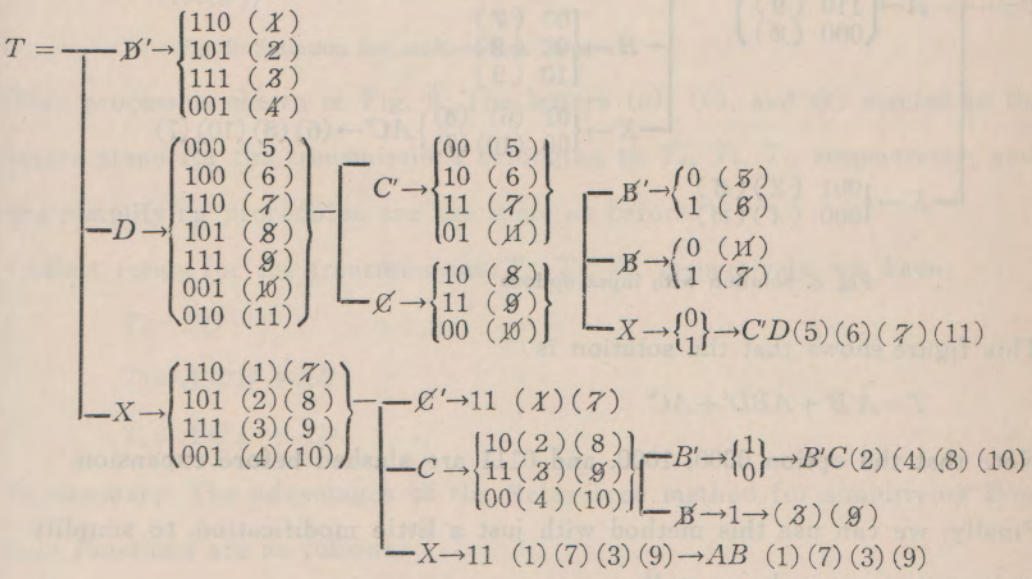


Fig. 7. Solution in the reverse direction.

The same results is obtained. This shows that the Relay-Tree Method is easily used in the reverse direction. But Scheinman's Method may cause trouble if used in the reverse direction.

In addition, if there are any optional input combinations, we just need to slash the designating numbers for those terms before expansion.

These optional imput-combinations are used only for obtaining the prime-implicants. Then we carry out the simplification procedures as before. (see Fig. 7)

Example 5.

$$T=0010+0001+0011+1001+1100+1101+1110,$$

with the input options

$$0000+1000+0111.$$

Here the figure will be

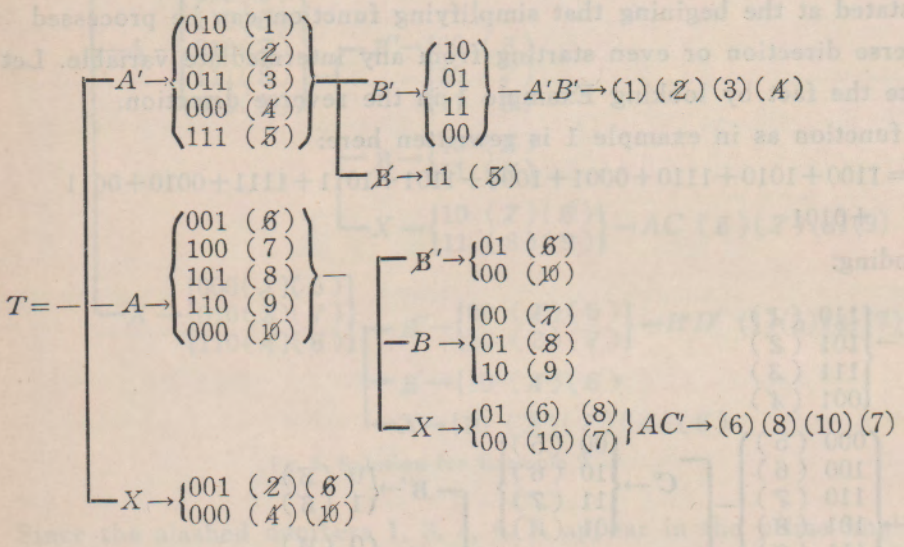


Fig. 8. Solution with input options

This figure shows that the solution is

$$T=A'B'+ABD'+AC'$$

Note that the option 0000, 1000, and 0111 are slashed before expansion.

Finally, we can use this method with just a little modification, to simplify multi-output networks as well.

Example 6

$$T_a=0101+0111+1101+1111$$

$$T_b=0000+0100+1100+1101+1110+1111$$

$$T_c=0000+0100+1001+1011+1101+1111$$

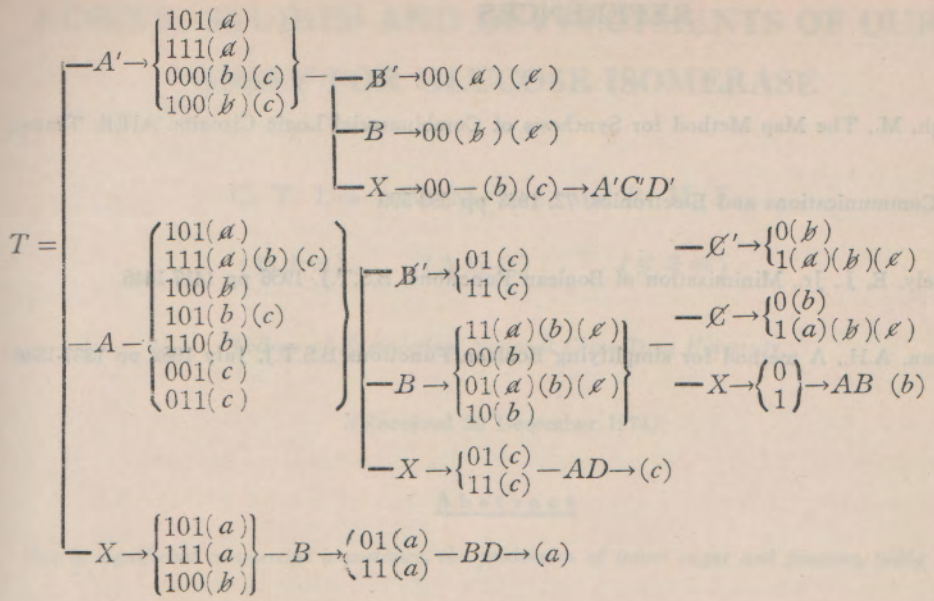


Fig. 9. Soliution for multi-output Networks

This process is shewn in Fig. 9. The letters (a), (b), and (c) circled in the figure stand for the transmissions belonging to T_a, T_b, T_c, respectively; and the simplifying procedures are the same as before.

Collect terms for the transmissions T_a, T_b, T_c, respectively, we have:

$$T_a = BD$$

$$T_b = A'C'D' + AB$$

$$T_c = A'C'D' + AD$$

In summary: The advantages of the Relay-Tree method for simplifying Boolean functions are as follows:

*It can be applied to functions with a large number of variables

*The procedures can be programmed for a computer.

*This method is simple and the simplifications may begin with any variable and proceed in either direction.

*Fewer mistakes occur

*Often there is no need to expand to the last variable in order to obtain the correct result.

REFERENCES

1. Karnaugh, M., The Map Method for Synthesis of Combinatorial Logic Circuits, AIEE. Trans., Part 1; Communications and Electronics, 72, 1953 pp 593-599

2. Mocluskely, E, J., Jr., Minimization of Boolean Functions, B,S.T.J. 1956 pp 1417-1445

3. Scheinman, A.H., A method for simplifying Boolean Functions B,S.T.J. July 1962 pp 1337-1346