Synthesis of TANT networks using a Boolean analyser*

By Miguel A. Marin†

The automated synthesis of TANT combinational networks (Three-level AND-NOT networks with True inputs) is studied using the advantages of Svoboda Boolean Analyzer (SBA) (see Svoboda, 1968). The synthesis procedure consists of (i) calculating the ordinary prime implicants of the given function f using operation Mode I of SBA, (ii) calculation of all generalised prime implicants of f by solving a system of Boolean equations (operation Mode II), (iii) solution of a covering problem using a special Petrick function and operation Mode I of the SBA. The synthesis procedure yields a TANT network containing a minimum number of AND-NOT elements. FORTRAN IV programs have been developed which implement the proposed synthesis method; however, the SBA hardware unit, when available, will prove an efficiency factor in processing time of approximately 10^4.

(Received November 1968)

The synthesis of Three-level AND-NOT combinational networks with True inputs (TANT networks) has been studied by Maley and Ogden (1962), McCluskey (1963), Hellerman (1963), Maley and Earle (1963) and recently by Gimpel (1967). This last author presents the first systematic synthesis and simplification method of TANT circuits. This method is very similar to the Quine–McCluskey minimisation algorithm for two-level AND-OR logic.

Gimpel's method starts by generating a general form for the implicants of the given Boolean function f, called *permissible-implicants* of f, and in such a way that it is simple to derive the circuit configuration from this general form. From the set of permissible implicants, the set of *prime-permissible implicants* (which we shall call here generalised prime implicants) is determined and from this set those which realise the given function with a minimum number of NAND decision elements are selected.

In this paper we propose another method of TANT synthesis based on the two modes of operation of Svoboda's Boolean Analyzer (Svoboda, 1968).

The proposed method consists of calculating the ordinary prime implicants of the given function f, and from this set to calculate all generalised prime implicants of f by solving a system of Boolean equations. The selection of those generalised prime implicants which form a minimal TANT network is done by constructing a special Petrick function (Petrick, 1956) Z and determining its prime implicants. The proposed method thus uses the two modes of operation of the Boolean Analyser: Mode II for the determination of the generalised prime implicants and Mode I in triadic mode for the determination of the minimal TANT networks.‡

‡ See Marin, 1968, for a detailed description of the two modes of operation of Svoboda's Boolean Analyzer.

Definitions and theorems

Consider the TANT circuit of **Fig. 1.** It is easily verified that this circuit implements the function

$$f = x_2x_0 + x_0\overline{(x_1x_0)} + x_1\bar{x}_2\overline{(x_0x_1)} \qquad (1)$$

This function is written in AND-OR form as follows:

$$f = x_2x_0 + x_0\bar{x}_1 + x_1\bar{x}_2\bar{x}_0.$$

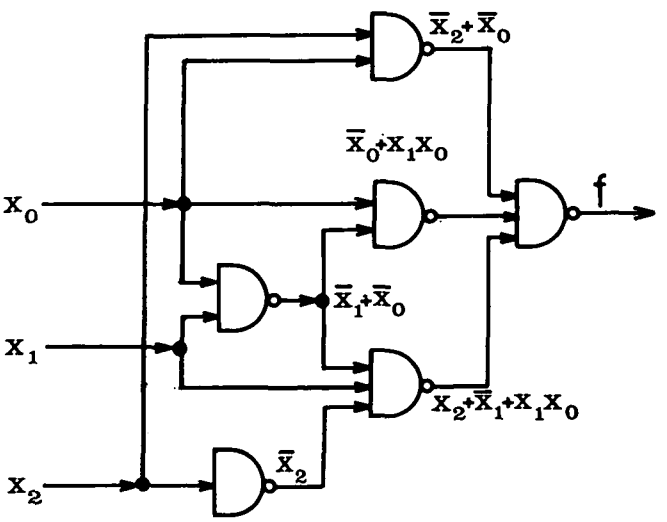


Fig. 1. Example of TANT circuit

Form (1) of f presents a characteristic structure of sum of permissible terms (P-terms) whose general form may be defined in the following way:

* This Research was done at the Department of Electrical Engineering, University of California at Los Angeles and is part of the author's Ph.D. dissertation.

† *Department of Electrical Engineering, McGill University, Montreal, Canada*

Definition 1

Every term P, implicant of a Boolean function f of n variables and expressed in TANT for (1), may be written as a product of two factors H and T

$$P = H \,.\, T \tag{2}$$

where

$H = \ddot{x}_{n-1} \ldots \ddot{x}_i \ldots \ddot{x}_0 \ddot{x}_1$ — $\ddot{x}_i$ takes the values 1 or x_i exclusively ($i = 0, \ldots, n-1$) and

$T = T_{m-1} T_{m-2} \ldots T_1 T_0$ — $T_j = \dot{x}_{n-1} + \ldots + \dot{x}_i + \ldots + \dot{x}_0$; $\dot{x}_i$ takes the values 0 or $\bar{x}_i$ exclusively. ($j = 0 \ldots m-1$), ($i = 0 \ldots n-1$), $m \leqslant n$.

Example 1

The term $P_1 = x_0\overline{(x_0 x_1)}$ has $H = x_0$,

$$T = T_1 = \overline{(x_0 x_1)} = (\bar{x}_1 + \bar{x}_0).$$

The term $P_2 = x_1 \bar{x}_2 \overline{(x_0 x_1)}$ has $H = x_1$,

and $T = T_2 \,.\, T_1 = \bar{x}_2(\bar{x}_0 + \bar{x}_1)$.

Definition 2

A term P of the form (2) is said to be a *generalised prime implicant* of f (*GP*-term) if an ordinary prime implicant of f (in the sense of the AND-OR logic) or a sum of any number of them is obtained when a P-term (2) is expanded using the rules of logical sum, product and complementation. The ordinary prime implicants of f will be denoted by OP.

Example 2

The function f given in Fig. 1 has the following ordinary prime implicants:

$$x_0\bar{x}_1,\ x_0x_2,\ \bar{x}_0x_1\bar{x}_2.$$

The terms P_1 and P_2 of example 1 corresponding to the TANT form (1) of f are generalised prime implicants. Effectively, according to definition 2,

$$P_1 = GP_1 = x_0\overline{(x_1x_0)} = x_0\bar{x}_1 + x_0\bar{x}_0 = x_0\bar{x}_1$$

$$P_2 = GP_2 = x_1\bar{x}_2\overline{(x_0x_1)} = x_1\bar{x}_2\bar{x}_1 + \bar{x}_2\bar{x}_0x_1 = \bar{x}_2\bar{x}_0x_1$$

Definition 3 (Criterion of minimal TANT network)

We shall say that a TANT network produces a certain given Boolean function f in a minimal form if no other TANT circuit exists which has less number of NAND decision elements.

Theorem 1

According to the previous definition of minimal TANT network, every P-term (2) of a minimal TANT form of f is a generalised prime implicant.

Proof:

Suppose that the minimal TANT expression of f is

$$P_1 + P_2 + \ldots + P_i + \ldots + P_r$$

where each term P_i is of the form (2). We shall prove that P_i is formed by logical addition of any number of ordinary prime implicants OP of f. To prove this statement it is sufficient to prove that there is no minterm m_i of f such that

$$P_i = \Sigma_j OP_j + m_i \text{ with } m_i \notin \Sigma_j OP_j.$$

Effectively, according to definition 1 the general form of P_i is $H_1(T_1T_2 \ldots T_k)$; on the other hand, since OP_j is an ordinary prime implicant of f,

$$\Sigma_j OP_j = H_1\bar{x}_1 \ldots \bar{x}_i \ldots \bar{x}_m,\ x_i \notin H_1$$

for $\qquad i = 1 \ldots m,\ m \leqslant n.$

But $P_i = \Sigma_j OP_j + m_i$; therefore, the minterm m_i should have the same factor H_1 of uncomplemented variables that the terms OP_j and P_1. But m_i is a minterm and thus contains all complemented variables which are not contained in H_1. These variables must be contained also in $\Sigma_j OP_j$ because, if not contained, or m_i is not a minterm or its H-factor of uncomplemented variables does not coincide with H_1, implying that P_i is not of the form (2). Therefore every term of a minimal TANT form is a generalised prime implicant.

The question is now how to determine the set of generalised prime implicants and to select from this set those that produce the function f according to definition 3.

To form the set of generalised prime implicants we start from the set of ordinary prime implicants of f as proven by the following theorem.

Theorem 2

Let $\{OP\}$ be the set of ordinary prime implicants of f and let $\{GP\}$ be the set of generalised prime implicants, then

$$\{OP\} \subset \{GP\}$$

$$GP_j = \Sigma_i OP_i \text{ for } OP_i \in \{OP\};\ GP_j \in \{GP\}.$$

The proof of these two propositions follows immediately from the definition of ordinary prime implicant of f and from definition 2.

Based on theorem 2, it is clear that the set of ordinary prime implicants of f augmented with those P-terms of the form (2), obtained by logical addition of any number of prime implicants OP_i contained in $\{OP\}$, constitutes the set of P-terms from which it is possible to generate all possible generalised prime implicants GP_j. This set of P-terms thus formed is called the BASE of the given function f. It will be denoted by B.

Note that once the set of ordinary prime implicants is obtained, only those with common H-factor will produce a new P-term of the base by logical addition. This is due to the special form of the P-terms (2).

Example 3

Consider the function of **Fig. 2.**

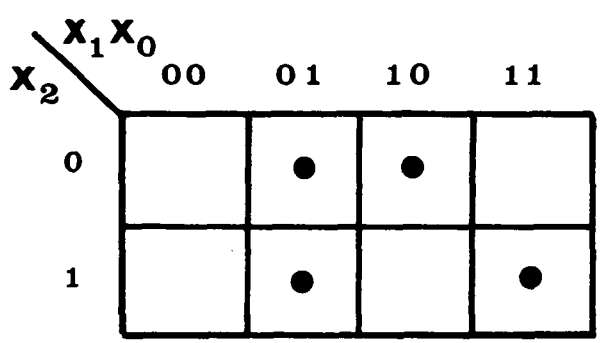


Fig. 2. TANT synthesis example

The set of ordinary prime implicants is

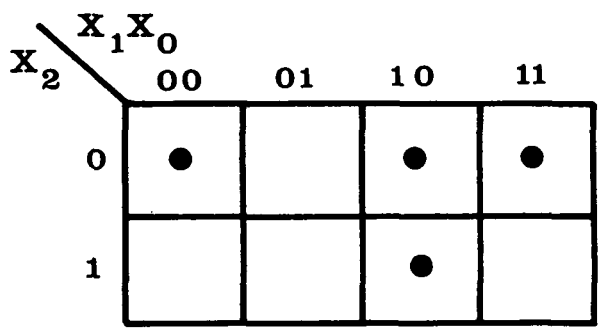
$$\{OP\} = \{x_0\bar{x}_1,\ x_0x_2,\ \bar{x}_0x_1\bar{x}_2\}.$$

Since there are no terms $OP_j \subset \{OP\}$ with common H-factor, the base B of the function coincides with the set $\{OP\}$. Thus

$$B = \{x_0\bar{x}_1,\ x_0x_2,\ \bar{x}_0x_1\bar{x}_2\}.$$

Example 4

Consider the function whose corresponding Marquand map is



The set $\{OP\}$ is $x_1\bar{x}_2$, $x_1\bar{x}_0$, $\bar{x}_0\bar{x}_2$. In this case there are two terms OP_j with common H-factor $x_1\bar{x}_2$ and $x_1\bar{x}_0$. These two terms are combined by logical addition producing the term $x_1\overline{(x_2x_0)}$. The BASE is thus

$$B = \{x_1\bar{x}_2,\ x_1\bar{x}_0,\ \bar{x}_0\bar{x}_2,\ x_1\overline{(x_2x_0)}\}.$$

Calculation of the generalised prime implicants

The generalised prime implicants of a given Boolean function f are constructed by inserting in all possible ways the variables contained in the H-factor of each term of the base in its corresponding T-factors. The GP terms thus obtained which are identically equal to zero are disregarded.

Example 5

Consider the base $B = \{x_0\bar{x}_1,\ \bar{x}_0x_1\bar{x}_2,\ x_0x_2\}$. From the term $x_0\bar{x}_1$ we obtain the term $GP_1 = x_0\overline{(x_0x_1)}$. From the term x_0x_2 no acceptable term is generated because those obtained: $x_0x_2(\bar{x}_0)$, $x_0x_2(\bar{x}_2)$, $x_0x_2\overline{(x_0x_2)}$ are identically equal to zero. From term $x_1\bar{x}_0\bar{x}_2$ we obtain

$$GP_2 = x_1\overline{(x_1x_0)}\bar{x}_2$$
$$GP_3 = x_1\bar{x}_0\overline{(x_2x_1)}$$
$$GP_4 = x_1\overline{(x_1x_0)}\overline{(x_2x_1)}.$$

The generalised prime implicants are:

$$GP_1 = x_0\overline{(x_1x_0)}$$
$$GP_2 = x_1\overline{(x_1x_0)}\bar{x}_2$$
$$GP_3 = x_1\overline{(x_2x_1)}\bar{x}_0$$
$$GP_4 = x_1\overline{(x_2x_1)}\overline{(x_0x_1)}$$

plus the P-terms of the base

$$GP_5 = x_0\bar{x}_1$$
$$GP_6 = \bar{x}_0x_1\bar{x}_2$$
$$GP_7 = x_0x_2$$

This method of calculating the generalised prime implicants is preferred when the synthesis is done *manually*. However, the automatisation by standard computer programming of this method is laborious because it is required to handle variable-length words and many shift operations to insert the variables of the H-factor in the T-factor. Also the identification of H- and T-factors belonging to the same P-term is not easy to implement. For these reasons and due to the fact that the Boolean Analyser is capable of solving large systems of Boolean equations, we studied the possibility of transforming the problem of generalised prime implicants generation to the solution of a system of Boolean equations. Before describing the set-up of this system of equations we need the following theorem.

Theorem 3

If a T-factor of a P-term belonging to the base B of a given Boolean function f of n variables has $m (m \leqslant n)$ complemented variables, the GP-terms obtained from this P-term has m distinct T_j factors ($j = 1 \ldots m$).

Proof:

Let the term $OP_1 = x_0 \overset{s}{\ldots} x_i \ldots \bar{x}_j \overset{r}{\ldots} \bar{x}_k$, where $(r + s) \leqslant n$, $x_0 \overset{s}{\ldots} x_i \neq x_j \overset{r}{\ldots} x_k$, and let $P = OP_j$, $P \subset B$ and $OP_j \subset \{OP\}$. Suppose that it is possible to generate from this P-term a generalised prime implicant of the form

$$GP_1 = x_0 \overset{s}{\ldots} x_i \cdot \bar{x}_j \overset{r}{\ldots} \bar{x}_k(T_q). \tag{3}$$

According to Theorem 2 the factor T_q has to have the form

$$T_q = (a\bar{x}_0 + b\bar{x}_1 + \overset{s}{\ldots} + h\bar{x}_i + A\bar{x}_j + \overset{r}{\ldots} + K\bar{x}_k)$$

(where $a, b, \ldots, h, A, \ldots, K$ are logical variables), because if it does not have this form, either P does not belong to the base B or GP_1 (3) is not a P-term of the form (2). But, suppose that the term $GP_1(3)$ is accepted as a generalised prime implicant of the *minimal* TANT expression, then it is evident that the circuit thus obtained will use an additional gate to synthesise T_q as compared with the circuit obtained by accepting the term P instead of GP_1. In the case that the factor T_q is available already in the synthesised circuit because it may belong to a necessary GP_j-term ($j \neq 1$), it is easily seen that the term P is also preferred instead of GP_1, because by using P an input at the decision element that realises the term GP_1 is saved. Therefore, the term GP_1 (3) with $m + 1$ factors T does not introduce any additional information in the synthesis process and thus is not considered as an effective generalised prime implicant of f.

Theorem 3 gives the general form of the equation to be solved to obtain the acceptable generalised prime implicants. Effectively, given a term P_i of the base,

$$P_i = x_0 \overset{s}{\ldots} x_i \cdot \bar{x}_j \overset{r}{\ldots} \bar{x}_k,\ (r + s) \leqslant n$$
$$x_0 \overset{s}{\ldots} x_i \neq \bar{x}_j \overset{r}{\ldots} \bar{x}_k$$

every GP_j-term derived from P_i, $P_i \in B$, is a solution of the equation

$$x_0 \overset{s}{\ldots} x_i \,.\, \bar{x}_j \overset{r}{\ldots} \bar{x}_k = x_0 \overset{s}{\ldots} x_i \,.\, (a_1\bar{x}_0 + b_1\bar{x}_1 + \overset{n}{\ldots} + h_1\bar{x}_{n-1}) \ldots (a_r\bar{x}_0 + b_r\bar{x}_1 + \overset{n}{\ldots} + h_r\bar{x}_{n-1}) \quad (4)$$

where a_q, b_q . . . h_q for $q = 1, 2, \ldots, r$ are logical variables.

The equation (4) may be considered as a system of Boolean equations with the unknowns a_q, b_q . . . h_q ($q = 1 \ldots r$). This equation may be written in the form $Y = 0$ as required for the Boolean Analyser. The discriminant is obtained using Mode II of the Boolean Analyser which is used to obtain all possible solutions of (4). The values of unknowns substituted in the right-hand side of (4) give the terms P which form the set of generalised prime implicants originated by the term P of the base.

Example 6

Consider the term $P_1 = x_1\bar{x}_0\bar{x}_2$. The set of generalised prime implicants derived from this term was already obtained in Example 5. The corresponding equation (4) for P_1 is:

$$x_1\bar{x}_0\bar{x}_2 = x_1(a_1\bar{x}_2 + b_1\bar{x}_1 + c_1\bar{x}_0)(a_2\bar{x}_2 + b_2\bar{x}_1 + c_2\bar{x}_0). \quad (5)$$

Let us form the discriminant of this equation, i.e. in the space of $2^6 \,.\, 2^3 = 2^9$ (6 unknowns and 3 constants) let us determine the minterms for which equation (5) is satisfied. This discriminant is shown in **Fig. 3**.

The interesting solutions to our problem are those which are invariant with respect to the values of the constants x_0, x_1, x_2; i.e. those which satisfy (5) irrespective of the values of the constants x_0, x_1, x_2. If the discriminant is organised as we have done in Fig. 3, rows representing the space of unknowns and columns the space of constants, the desired solutions may be read directly from the discriminant because each solution corresponds to every row containing only dots (a dot means that for this minterm equation (5) will hold). The solutions of (5) have been marked with an arrow in Fig. 3 for easy identification, and are listed as follows:

IDENTIFIER	SPACE OF UNKNOWNS	TERMS OBTAINED BY SUB. IN (5)
	$c_2b_2a_2c_1b_1a_1$	
12	0 0 1 1 0 0	$x_1(\bar{x}_0)(\bar{x}_2)$
14	0 0 1 1 1 0	$x_1(\overline{x_0x_1})(\bar{x}_2)$
28	0 1 1 1 0 0	$x_1(\bar{x}_0)(\overline{x_2x_1})$
30	0 1 1 1 1 0	$x_1(\overline{x_0x_1})(\overline{x_2x_1})$
33	1 0 0 0 0 1	$x_1(\bar{x}_2)(\bar{x}_0)$
35	1 0 0 0 1 1	$x_1(\overline{x_1x_2})(\bar{x}_0)$
49	1 1 0 0 0 1	$x_1(\bar{x}_2)(\overline{x_1x_0})$
51	1 1 0 0 1 1	$x_1(\overline{x_0x_1})(\overline{x_1x_2})$

From this set of solutions we eliminate those that are repeated because the logical multiplication is commutative. Thus solution 12 is identical to solution 33, 14 to 49, 28 to 35 and 30 to 51. Finally we obtain the following generalised prime implicants:

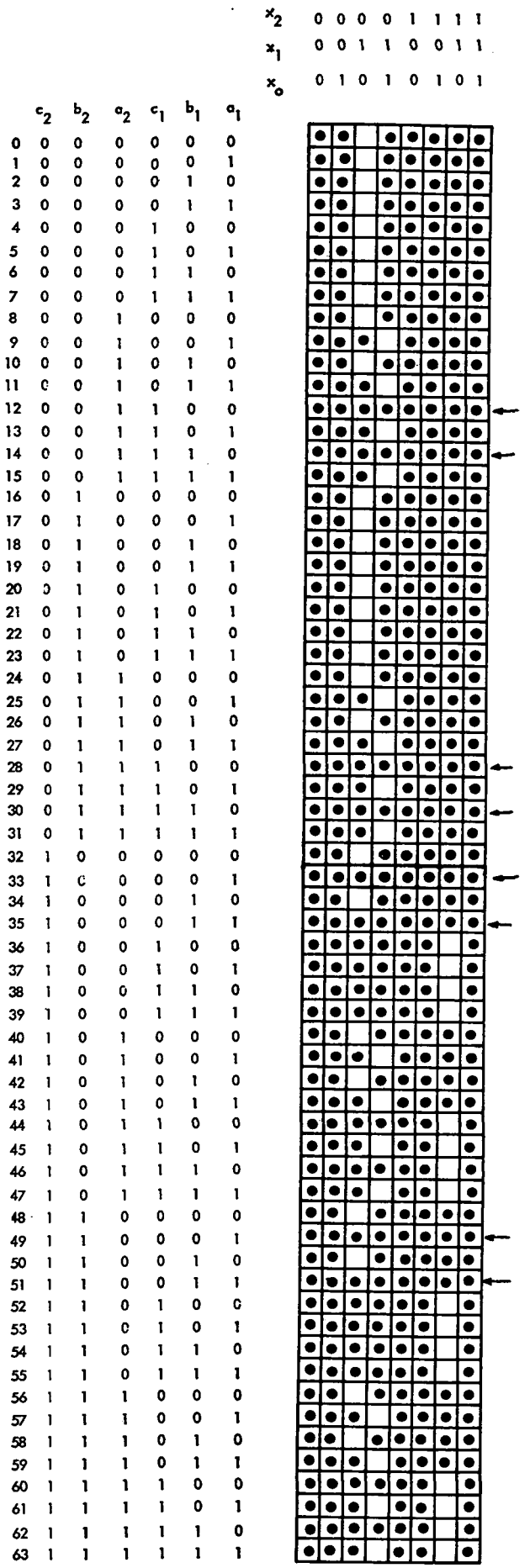


Fig. 3. Discriminant of equation (5)

$$GP_2 = x_1(\overline{x_1x_0})\bar{x}_2$$
$$GP_3 = x_1\bar{x}_0(\overline{x_2x_1})$$
$$GP_4 = x_1(\overline{x_0x_1})(\overline{x_2x_1})$$
$$GP_6 = x_1\bar{x}_0\bar{x}_2$$

which coincide with those previously obtained. (We have kept here the same sub-indices as in Example 5.)

The method of generating the generalised prime implicants through the solution of systems of Boolean equations is very long if done manually. Note, for example, that for functions with only 3 variables a logical space of 9 variables is required. The method, however, is completely general and is specially adapted to the Boolean Analyser (see Marin, 1968). The general purpose computer (in our case the Sigma 7 computer) will generate the terms of the base B, and for each term of this base, it will generate an equation like (4) but in the form $Y = 0$. The terms of Y will be inserted in the logical operation unit of the Boolean Analyser and the discriminant will be obtained in the circulating memory (in our case a RAD Storage System). Sigma 7 will read out this discriminant selecting those identifiers from the space of unknowns which verify (4). The solutions thus obtained are stored in the memory of Sigma 7 and the system proceeds with the next term of the base. At the end of this process, all generalised prime implicants of f will be stored in Sigma 7.

In order to show the automatisation possibility of the above described method, a program was derived for Sigma 7, called SBEV3 (Systems of Boolean Equations Version 3) which simulates the Boolean Analyser in this particular application. The program SBEV3 was written in FORTRAN IV and for functions f of three variables only.* A full listing of this program and the output data obtained for the function f of three variables† is available on application to the author.

To interpret properly the output data given by the program SBEV3, the following terminology is used:

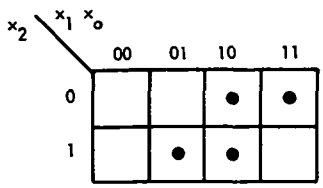
Base term number	Refers to the ordering number of the base
List of terms of the function $Y = 0$	Refers to the terms of equation (4) written in form $Y = 0$
Identifier of the solution	Refers to the identifier of the logical space of unknowns for which a solution exists.

In our case, since there are 3 unknowns, this logical space corresponds to the unknowns a, b, c, exclusively and in correspondence with the constants x_2, x_1, x_0 respectively. Thus, for the first term of the base (see **Table 1**), $x_0x_2\bar{x}_1$, the program SBEV3 gives the following solutions:

Identifiers	*a,b,c*	*T-Factor*	*GP-Term*
2	0 1 0	$\bar{x}_1$	$x_2x_0\bar{x}_1$
3	0 1 1	$(\overline{x_1x_0})$	$x_2x_0(\overline{x_1x_0})$
6	1 1 0	$(\overline{x_2x_1})$	$x_2x_0(\overline{x_2x_1})$
7	1 1 1	$(\overline{x_2x_1x_0})$	$x_2x_0(\overline{x_2x_1x_0})$

For the second term of the base, $x_1\bar{x}_0$ the solutions are

* A more general program is presented in Marin, 1968, Chapter 5.
† This function (shown above Table 1) is taken from an example presented by Gimpel.



Generalized Prime Implicants		
Terms of the Base		Calculated GP Terms
Number	Expression	
1	$x_2x_0\bar{x}_1$	$x_2x_0(\overline{x_1x_0})$ $x_2x_0(\overline{x_1x_2})$ $x_2x_0(\overline{x_0x_1x_2})$
2	$x_1\bar{x}_0$	$x_1(\overline{x_0x_1})$
3	$x_1(\overline{x_0x_2})$	$x_1(\overline{x_0x_1x_2})$
4	$x_1\bar{x}_2$	$x_1(\overline{x_1x_2})$

Table 1

Generalised prime implicants

1	0 0 1	$\bar{x}_0$	$x_1\bar{x}_0$
3	0 1 1	$(\overline{x_1x_0})$	$x_1(\overline{x_1x_0})$
5	1 0 1	$(\overline{x_2x_0})$	$x_1(\overline{x_2x_0})$
7	1 1 1	$(\overline{x_2x_1x_0})$	$x_1(\overline{x_2x_1x_0})$

For the fourth term of the base, $x_1\bar{x}_2$ the solutions are

4	1 0 0	$\bar{x}_2$	$x_1\bar{x}_2$
6	1 1 0	$(\overline{x_2x_1})$	$x_1(\overline{x_2x_1})$

The prime implicants obtained with the program SBEV3 coincide with those of Table 1.

The selection of generalised prime implicants

In order to obtain a minimal TANT synthesis of a Boolean function f, not all generalised prime implicants are required. This is also the case of the AND-OR logic of two levels. The problem now is to select those generalised prime implicants which produce the function f according to the criterion of minimality given in definition 3.

In a TANT expression and due to the fact that there exists a third logical level, it is possible to use the same elements of the third logical level in several inputs of elements of the other levels and therefore the selection of *GP*-terms consists of two problems:

1. The selection of the minimum number of *GP*-terms which cover the function.
2. From all TANT expressions obtained select those which use the least number of NAND elements, i.e. obtain maximum sharing of *T*-factors.

The first problem is a classical one and has been solved by McCluskey (1965) using tables or algebraically by Petrick (1956) by means of a logical function Z which expresses the condition of covering of the function f.

The second problem has been solved by Luccio and Grasselli (1965) using a special table called a CC-table (Cover and Closure Table) but in connection with the problem of simplification of the number of internal states of a sequential circuit. In this section we propose a similar method as the one using CC-tables but we differ in the fact that our table is used to generate two special Petrick functions that in combination solve the covering problem.

In the following lines we describe systematically our investigations in solving this covering problem.

We apply first the Petrick function to the example on Table 1. For this purpose a table is constructed whose columns represent the minterms of the function f and the rows the generalised prime implicants. We enter a dot in those places of the table where the generalised prime implicant of a row covers a minterm of a column. See **Table 2.**

	$x_2x_1x_0$	$x_2\bar{x}_1x_0$	$x_2x_1\bar{x}_0$	$x_2\bar{x}_1\bar{x}_0$
$GP_1 = x_2x_0\bar{x}_1$			•	
$GP_2 = x_2x_0\overline{(x_1x_0)}$			•	
$GP_3 = x_2x_0\overline{(x_1x_2)}$			•	
$GP_4 = x_2x_0\overline{(x_2x_1x_0)}$			•	
$GP_5 = x_1\bar{x}_0$	•			•
$GP_6 = x_1\overline{(x_1x_0)}$	•			•
$GP_7 = x_1\overline{(x_2x_0)}$	•	•		•
$GP_8 = x_1\overline{(x_2x_1x_0)}$	•	•		•
$GP_9 = x_1\bar{x}_2$	•	•		
$GP_{10} = x_1\overline{(x_2x_1)}$	•	•		

Table 2
Covering of minterms

If the proposition 'All minterms of the function are covered' is identified with the Boolean variable Z, this variable may be expressed as a function of the GP_i ($i = 1 \ldots 10$) terms as follows:

$$Z = (GP_5 + GP_6 + GP_7 + GP_8 + GP_9 + GP_{10}) \cdot (GP_7 + GP_8 + GP_9 + GP_{10}) \cdot (GP_1 + GP_2 + GP_3 + GP_4) \cdot (GP_5 + GP_6 + GP_7 + GP_8) \quad (6)$$

Each term gives us a covering of f. Evidently those terms of Z with the least number of GP-terms will give the most economical implementation. In other words, the ordinary prime implicants of Z are the irredundant coverings of f. The expansion of (6) gives

$$Z = GP_1 \cdot GP_7 + GP_2 \cdot GP_7 + GP_3 \cdot GP_7 + GP_4 \cdot GP_7 + GP_1 \cdot GP_8 + GP_2 \cdot GP_8 + GP_3 \cdot GP_8 + GP_4 \cdot GP_8 \quad (7)$$

where each term is already a prime implicant of Z.

Any term of (7) that we may choose for the TANT circuit has the same number of GP-terms. This selection, however, does not suffice to obtain an implementation with a minimum number of gates, because it is possible to use an output of the third logical level (T-factor) as inputs to several gates of the second level. Inspection of Table 2 shows immediately that GP_4 and GP_8 have the same T-factor: $\overline{(x_0x_1x_2)}$ and since $GP_4 \cdot GP_8$ is a term of (7) these two GP-terms give a minimal TANT implementation of f. The corresponding circuit is given in **Fig. 4.**

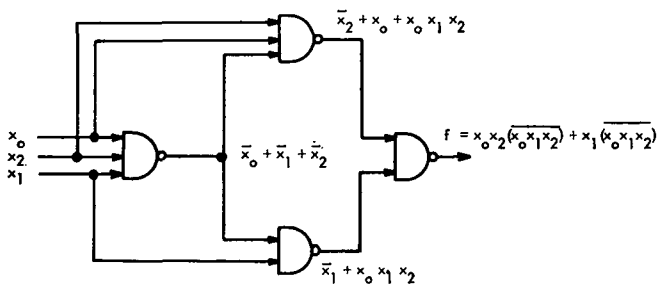


Fig. 4. TANT circuit corresponding to Table 1

The visual inspection of Table 2 is not in general so easy for problems of large numbers of variables, and since we are interested in automatic methods of selection, we investigate the following two alternatives:

1. to construct another table based on a function like (7)
2. to add logical conditions to the Petrick function to include the second selection process.

	$\bar{x}_1$	$\overline{(x_1x_0)}$	$\overline{(x_1x_2)}$	$\overline{(x_2x_1x_0)}$	$\overline{(x_2x_0)}$
$GP_1 \cdot GP_7$	•				•
$GP_2 \cdot GP_7$		•			•
$GP_3 \cdot GP_7$			•		•
$GP_4 \cdot GP_7$				•	•
$GP_1 \cdot GP_8$	•			•	
$GP_2 \cdot GP_8$		•		•	
$GP_3 \cdot GP_8$			•	•	
$GP_4 \cdot GP_8$				•	

Table 3

Let us consider both alternatives separately.

The *first alternative* consists of constructing another table (**Table 3**) in which each row corresponds to each term of the Petrick function previously obtained (7) and

with each column corresponding to every T_j factor of the *GP*-terms contained in (7). If a dot is entered in those places of the tables where a T_j-factor is present in a *GP*-term, the answer to our selection problem is the row(s) with the least number of dots. If several rows exist with the least number of dots, all give a minimal solution according to definition 3. However, if the criterion of minimum total number of inputs to the gates of the circuit is considered, we will choose from the table those T_j factors with the least number of literals.

To illustrate this process, consider the previous example and let us build a table (Table 3) in the way indicated with the terms of (7).

The row with the least number of points is the last one which corresponds to the term $GP_4 \,.\, GP_8$ as expected.

We now present another example of synthesis of a function of 4 variables.

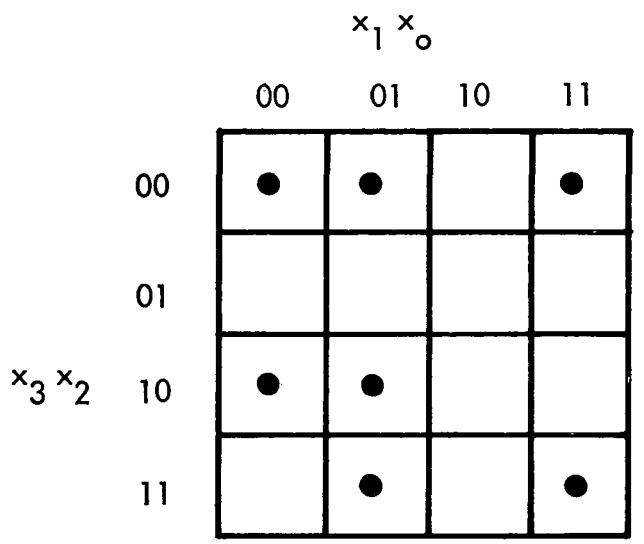
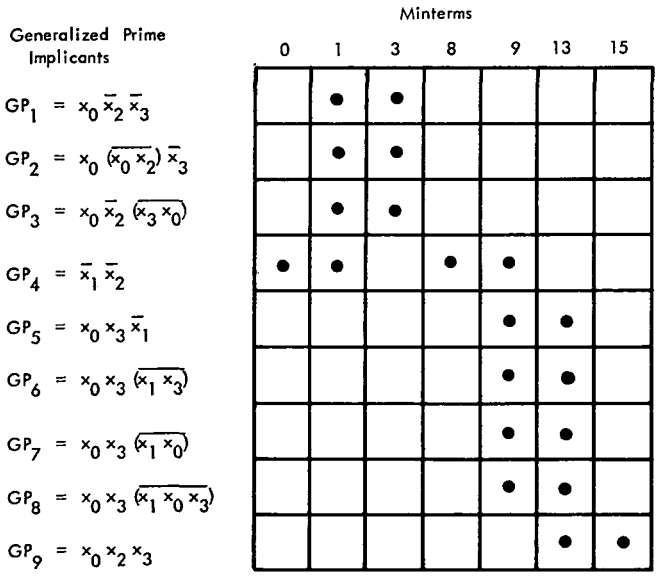


Fig. 5



Example 7

Let f be taken as given by the map of **Fig. 5.** The terms of the base are:

$$B = \{x_0\bar{x}_2\bar{x}_3,\ \bar{x}_1\bar{x}_2,\ x_0x_3\bar{x}_1,\ x_0x_2x_3\}$$

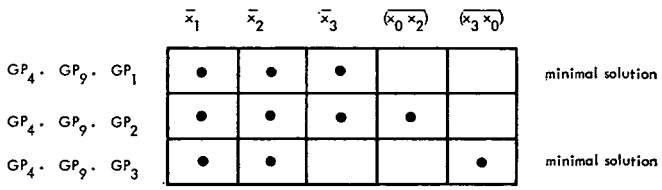
The corresponding Petrick function is:

$$\begin{aligned} Z = {} & GP_4(GP_1 + GP_2 + GP_3 + GP_4) \,. \\ & (GP_1 + GP_2 + GP_3) \,.\, GP_4 \,. \\ & (GP_4 + GP_5 + GP_6 + GP_7 + GP_8) \,. \\ & (GP_5 + GP_6 + GP_7 + GP_8 + GP_9) \,.\, GP_9 \end{aligned}$$

which is reduced to:

$$\begin{aligned} Z &= GP_4 \,.\, GP_9(GP_1 + GP_2 + GP_3) \\ &= GP_4GP_9GP_1 + GP_4GP_9GP_2 + GP_4GP_9GP_3 \end{aligned}$$

The associated table to this function is:



There exists two solutions with the same number of NAND elements:

$$f = GP_4 + GP_9 + GP_1 = \bar{x}_1\bar{x}_2 + x_0x_2x_3 + x_0\bar{x}_2\bar{x}_3 \tag{8}$$

$$f = GP_4 + GP_9 + GP_3 = \bar{x}_1\bar{x}_2 + x_0x_2x_3 + x_0\bar{x}_2\overline{(x_3x_0)} \tag{9}$$

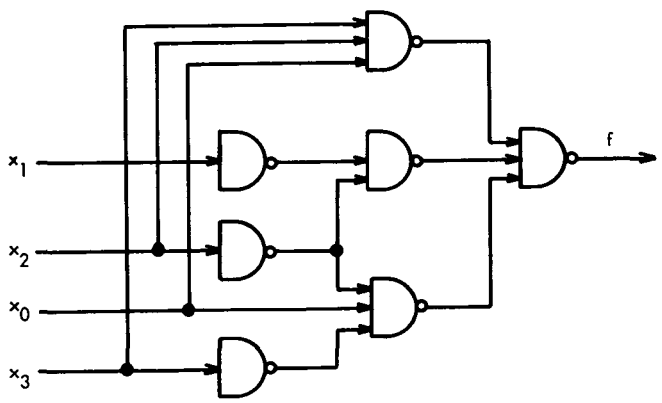


Fig. 6. TANT circuit corresponding to example 7, first solution

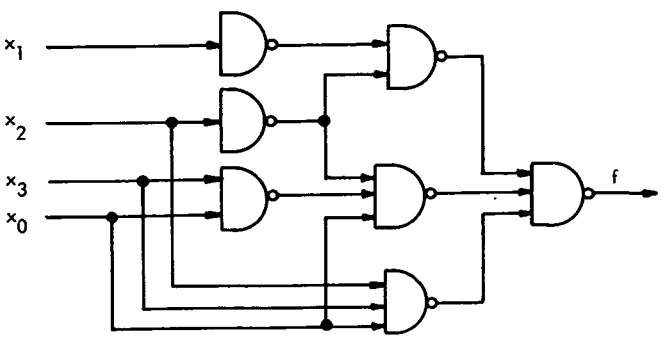


Fig. 7. TANT circuit corresponding to example 7, second solution

In **Figs. 6** and 7 we show the TANT circuits realising forms (8) and (9).

The *second alternative* in the selection of *GP*-terms is to add to the Petrick function Z a set of logical conditions (equations) that together with Z give the solution of the problem.

If we expand the original Table 1 from which Z (7) was derived by adding new entries each one corresponding to a T_j-factor belonging to every *GP*-term, and if a dot is used to mark those places of the expanded side when the T_j-factor belongs to a certain *GP*-term (as was done in Table 3), new equations may be generated that together with Z express the logical conditions for *minimal* covering.

This process was illustrated in the previous example. The extended table of *GP*-terms with the T_j-factors is given in **Table 4.** From the columns of minterms one deduces the Petrick function Z which is the same as (7) and by using the columns of T_j-factors one obtains the logical equations which express logically that the indicated T_j-factors belong to the corresponding *GP*-terms. The T_j-factors are named here with the letter Q (see Table 4). Thus the set of equations obtained is:

$$Z = GP_4GP_9GP_1 + GP_4GP_9GP_2 + GP_4GP_9GP_3 \quad (9)$$

$$\left.\begin{aligned} GP_1 &= Q_2 \cdot Q_3 & GP_5 &= Q_1 \\ GP_2 &= Q_3 \cdot Q_4 & GP_6 &= Q_6 \\ GP_3 &= Q_2 \cdot Q_5 & GP_7 &= Q_7 \\ GP_4 &= Q_1 \cdot Q_2 & GP_8 &= Q_8 \end{aligned}\right\} \quad (10)$$

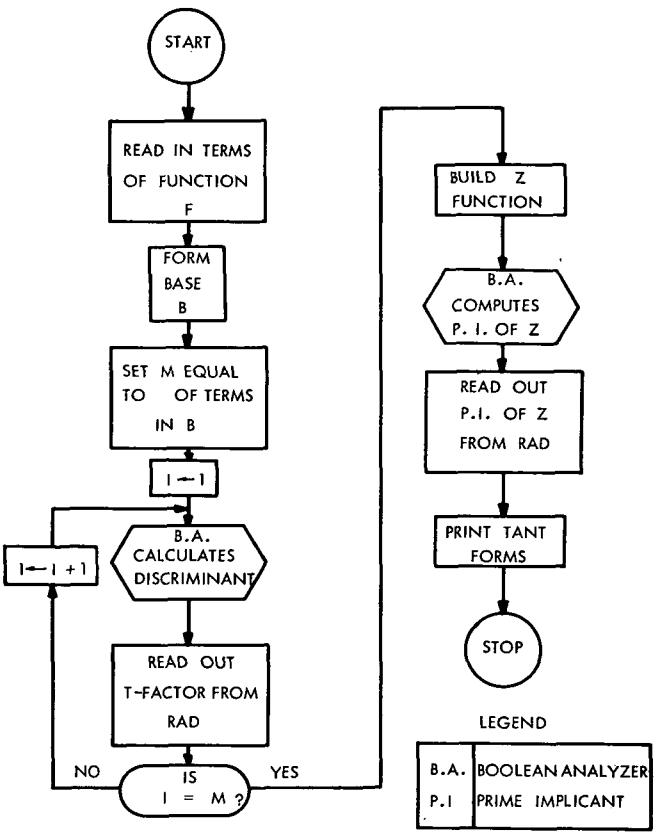


Fig. 8. Flowchart of the TANT synthesis using the Boolean analyser

Generalized Prime Implicants	Minterms 0	1	3	8	9	13	15	$\bar{x}_1$	$\bar{x}_2$	$\bar{x}_3$	$(\overline{x_2 x_0})$	$(\overline{x_3 x_0})$	$(\overline{x_3 x_1})$	$(\overline{x_1 x_0})$	$(\overline{x_1 x_0 x_3})$
$GP_1 = x_0 \bar{x}_2 \bar{x}_3$		●	●						●	●					
$GP_2 = x_0 (\overline{x_0 x_2}) \bar{x}_3$		●	●							●	●				
$GP_3 = x_0 \bar{x}_2 (\overline{x_3 x_0})$		●	●						●			●			
$GP_4 = \bar{x}_1 \bar{x}_2$	●	●		●	●			●	●						
$GP_5 = x_0 x_3 \bar{x}_1$					●	●		●							
$GP_6 = x_0 x_3 (\overline{x_1 x_3})$					●	●							●		
$GP_7 = x_0 x_3 (\overline{x_1 x_0})$					●	●								●	
$GP_8 = x_0 x_3 (\overline{x_1 x_0 x_3})$					●	●									●
$GP_9 = x_0 x_2 x_3$						●	●								
								Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8

Table 4
Selection table

If (10) is substituted into (7), the result obtained is:

$$Z = Q_1Q_2GP_9Q_3 + Q_1Q_2Q_3Q_4GP_9 + Q_1Q_2Q_5GP_9.$$

The first and the last terms contain the same least number of variables, and therefore are the solutions of the covering problem. Thus,

$$f = GP_4 + GP_9 + GP_1$$

$$f = GP_4 + GP_9 + GP_3$$

which coincide with (8) and (9) previously obtained.

From the point of view of automation, the second alternative seems to be more convenient because once the ordinary prime implicants of Z are obtained it suffices to do the above mentioned substitution (i.e. equations (10) in (g)) and this process may be done easily in the general purpose computer (Sigma 7).

In **Fig. 8** we show a schematic flow chart of the proposed automated process of TANT synthesis as carried out by the system Sigma 7—Boolean Analyser.

Svoboda's Boolean Analyzer has been simulated in a computer (Marin, 1968). This simulator, together with additional programs, are being developed to implement the synthesis process proposed here. However, the parallel processing feature of Svoboda's Boolean Analyzer hardware unit when available is expected to provide an efficiency factor of 10^4 when compared with an all-software implementation of this synthesis method.

References

MALEY, G. A., and OGDEN, S. (1962). *Minimal Three-Level NAND Circuitry*, Data Systems Division, IBM Corp., TROO.933.

MCCLUSKEY, E. J. (1963). Logical Design Theory of NOR Gate Networks with no Complemented inputs, *Proceedings of the 4th Annual Symposium on Switching Circuit Theory and Logical Design*, Chicago, Illinois, pp. 137–148.

HELLERMAN, L. (1963). A Catalog of Three-Variable OR INVERT and AND-INVERT Logical Circuits, *IEEE Trans. on Electronic Computers*, EC–12, pp. 198–223.

MALEY, GERALD A., and EARLE, JOHN (1963). *The Logical Design of Transistor Digital Computers*, Prentice-Hall.

GIMPEL, J. F. (1967). The Minimisation of TANT Networks, *IEEE Trans. on Electronic Computers*, Vol. EC–16, pp. 18–38.

SVOBODA, A. (1968). Boolean Analyzer, *Proc. of IFIP Conference*, 1968.

PETRICK, S. R. (1956). A Direct Determination of the Irredundant Forms of a Boolean Function from the set of Prime Implicants, Report AFCRC–TR–56–110, Cambridge, Mass.

MARIN, M. (1968). Investigation of the Field of Problems for the Boolean Analyzer, Ph.D. Dissertation, Department of Electrical Engineering, UCLA, June 1968.

MCCLUSKEY, EDWARD J. (1965). *Introduction to the Theory of Switching Circuits*, McGraw–Hill, New York.

GRASSELLI, A., and LUCCIO, F. (1965). A Method for Minimizing the Number of Internal States in Incompletely Specified Sequential Networks, *IEEE Trans. on Electronic Computers*, Vol. EC–14, pp. 350–359.

Book Review

Computer Evaluation of Mathematical Functions, by C. T. Fike, 1968; 227 pages. (Prentice-Hall, £5 5s. 0d. (cloth))

The word 'evaluation' in the title is short for 'evaluation and approximation', since of the eleven chapters more than half are directly concerned with the determination of polynomial and rational approximations to more complicated functions. Chapters 4, 5 and 7 have the respective titles 'Polynomial evaluation methods', 'Minimax polynomial approximation', 'Various polynomial approximation methods', and these are paralleled in Chapters 8–10 where 'polynomial' is replaced by 'rational function'. Chapter 6 deals with Chebyshev polynomials and series, and Chapter 11 with 'Asymptotic expansions'. Chapter 1 discusses the effect of rounding and data and truncation errors in the evaluation of functions, and some ways of assessing the error. Chapter 2 is devoted to iterative methods for evaluating square roots and cube roots, and Chapter 3, on 'Reducing the argument range', discusses some methods, reasons and dangers of performing this operation. There are numerous worked examples, over 150 exercises for the reader, and a wealth of bibliographical material about methods, codes and programs, not only from standard books but also from computing centres and computer firms (and for various machines) which are not usually mentioned in the standard literature.

One can criticise in very few places. I would have liked a little more on 'backward error analysis' in Chapter 1, and a reference somewhere to Moore's interval arithmetic; I am not persuaded by the arguments given for the accuracy of 'nested multiplication' or of the value of its combined use with the more economic methods of evaluating a polynomial; I would have liked a comment on why ill-conditioning in the solution of the linear equations involved in the Remez algorithms is dangerous; I would have liked a little more detail on the choice of the relative degrees of the polynomials in a rational approximation, and on the motivation for the method of economising continued fractions (p. 197), the formulae for once appearing out of the blue without supporting reasons; and I would have liked to see a comparison between the method of the text for improving the accuracy obtainable from asymptotic series with the almost forgotten 'convergence-factor' method of Bickley and Miller (Phil. Mag. **22**, 784, 1936) and Airey (Phil. Mag. **24**, 522, 1937).

But altogether this is an excellent book, a 'must' for computing centres, numerical and other mathematicians, and scientists with problems to solve. Almost all known valuable methods are included and illustrated by numerical example.

Continued on page 272