A MINIMIZATION ALGORITHM FOR TERNARY SWITCHING FUNCTIONS

T.C. Yang and A.S. Wojcik
Department of Computer Science
Illinois Institute of Technology
Chicago, Illinois

Abstract

An algorithm for the minimization of ternary switching functions is described. The algorithm is based on a standard binary switching function simplification method. Systematic methods for finding all prime implicants and a minimal-cost, static-and-dynamic-hazard-free representation for operation-hazard-free ternary switching functions are presented which are suitable for both computer and hand execution. Illustrative examples are also presented.

I. Introduction

Since the work of Post [1] in 1920, the development of multiple-valued logic has drawn the attention of many researchers and consequently, a fairly large number of papers have been published on this topic. In this paper we attempt to solve the following problem:

> Given an operation-hazard-free ternary switching function, define an algorithm to find a minimal-cost representation of the function which is free from static and dynamic hazards.

According to [2], an asynchronous network free of static and dynamic hazards can be designed for a given operation-hazard-free (OHF) function, f, which by definition is a function satisfying the relation

$$/ f(\vec{X}_i) - f(\vec{X}_y) / \leq 1 \qquad (1)$$

where $\vec{X}_i = (X_{1i}, X_{2i}, \ldots, X_{mi}, \ldots, X_{ki})$ and $\vec{X}_y = (X_{1y}, X_{2y}, \ldots, X_{my}, \ldots, X_{ky})$ are two adjacent k-tuples in which every $X_{mi}, X_{my} \in (0,1,\ldots,m-1)$ for m-valued logic. $\vec{X}_i$ and $\vec{X}_y$ are adjacent if and only if they differ in at most one component X_{hi} and X_{hy}, $1 \leq h \leq k$, and $/X_{hi} - X_{hy}/ \leq 1$. A general statement of the design process is also given in [2]. The basic idea is to introduce redundancy to mask transitional function values and therefore to prevent static and dynamic hazards in the fundamental mode of asynchronous operation, i.e. only one input is allowed to change at a time. A simple function is given in Table 1 to illustrate this process. The functional representation of this table is 1.(B,C,E,F) + 2.(C) which is free from hazards because the transition between B and C, B and E, C and F, and E and F are prevented by the term 1.(B,C,E,F). This characteristic can be further illustrated by the transition between B and C. Since the term 2.(C) is to produce a functional value 2, the term 1.(B,C,E,F) will also produce the functional value 1. This redundancy masks the transitional values of the term 2.(C) that produces the value 2 for C as the term changes from 0 to 2 and thus prevents static and dynamic hazards.

One of the major constraints for this minimization algorithm is the requirement that it results in a functional representation free of static and dynamic hazards. Our technique applies a specially designed covering algorithm that satisfies this constraint. One distinguishable characteristic of this algorithm is its reliance on the use of the join and complement join operation, to be defined in section III, to reduce the cost of the final solutions. As far as the cost is concerned, only the factor of the number of input building blocks will be taken into account. Other factors, such as speed of operation etc., will not be considered.

Section II gives a brief review of ternary switching algebra. Section III presents some basic definitions and two useful theorems. The minimization algorithm and its implementation are introduced in section IV. The process of minimizing the representation for selected function is used as a final example in section V. Finally the proofs of theorems are presented in the appendix.

II. A TERNARY SWITCHING ALGEBRA

Consider a ternary algebraic system having a set of elements 0, 1, and 2. Defining the Postian operations of addition (+) and multiplication (.) on L = (0,1,2) as

$$Y + X = X + Y = \text{MAX } (X,Y) \qquad (2)$$
$$Y \,.\, X = X \,.\, Y = \text{MIN } (X,Y)$$

yields a distributive lattice (L, +, .) with zero element 0 and universal element 2.

Let us define unary operators as follows:

$$X^{ijk} = \begin{cases} k & \text{if } X = 0 \\ j & \text{if } X = 1 \\ i & \text{if } X = 2 \end{cases} \quad \text{where } i,j,k \in (0,1,2)$$

there exists $3^3 = 27$ different unary functions on L to L. The unary functions to be used in this paper are listed in Table 2.

The following properties of ternary algebra, stated in terms of the notation to be used throughout this paper, are useful:

$X^X = X$
$X^{\bar{X}} = \overline{X} = (m-1)-X$ (m = 3 for ternary switching algebra)
$X^i = X^{i_2 i_1 i_0} = i_X$
$X^{002} = X^{020} + X^{200} = X^{222}$
$X^{020} = X^{022} \,.\, X^{220}$

Denoting the nth Cartesian power of L as L^n, we note that any ternary function of n variables, $X_1, X_2, \ldots, X_n$, is a mapping from L^n to L. Any such function may be represented by the expansion theorem:

This work was supported by the National Science Foundation, Division of Computer Research, under Grant DCR74-01409.

$$f(X_1, X_2, \ldots, X_n) = \sum_{\alpha} f(a_1, a_2, \ldots, a_n).X_1^{a_1}X_2^{a_2}\ldots X_n^{a_n} \quad (3)$$

for all $\alpha = (a_1, a_2, \ldots, a_n)$ in L^n, where $a_i \in (0,1,2)$ or $f(X_1,X_2,\ldots,X_n) = \sum_{f(\alpha)=2} X_1^{a_1} X_2^{a_2} \ldots X_n^{a_n} + 1.\sum_{f(\alpha)=1} X_1^{a_1} X_2^{a_2} \ldots X_n^{a_n} = g(X_1,X_2,\ldots,X_n)+1.h(X_1,X_2,\ldots,X_n) \quad (4)$

Equation (4) is called the canonical sum-of-products form of f. Both g and h are functions that attain the values 0 or 2, and the term 1 . h is thus either 0 or 1.

III. FUNCTION SIMPLIFICATION

Techniques adapted from standard binary minimization procedure will be described for obtaining a minimal-cost sum-of-product representation of a given function. There are two categories of binary minimization techniques, namely map methods and algebraic methods. Although map methods provide a convenient approach to logic design, their applicability in ternary case is limited in practice, due to the large number of combinations, to functions of up to three variables [2,3]. We shall now discuss a method which is an adaption of Quine's binary procedure [4]. The method suggested is systematic and can be easily mechanized.

Let us start by defining some basic terms that are modifications of their binary equivalents and basically are an extension of [5].

Definition 1: A constant, $r \in (0,1,2)$ is a value of a ternary variable.

Definition 2: $X_i^{a_i}$ is defined to be a literal, where $a_i \in (x,\bar{x},002,020,022,220,200,222)$

For n = 3, some product terms are:

(1) $1 \cdot X_1^{x} X_2^{\bar{x}} X_3^{x}$

(2) $2 \cdot X_1^{020} \cdot X_2^{200} = 2.X_1^{020} X_2^{200} X_3^{222}$

(3) $X_1^{022} X_2^{220} X_3^{012} = 2.X_1^{022} X_2^{220} X_3^{012}$

Definition 4: A product term with constant 1 is called a 1-term, and a product term with constant 2 is called a 2-term.

Definition 5: The i^{th} coordinate of a product term is defined to be the i^{th} variable from left in that term. For example, if $T = 1 \cdot X_1^{022} X_2^{220} X_3^{012}$, then the 3rd coordinate indicates X_3.

Definition 6: Literals of the form X_i^{x} or $X_i^{\bar{x}}$ are are called type 2 literals, all other literals are called type 1.

Definition 7: The index, K, of a product term is the number of type 2 literals which it contains.

Definition 8: A product term, T_1, subsumes another product term, T_2, written $T_1 \leq T_2$, iff $T_1(\alpha)$ is less than or equal to $T_2(\alpha)$ for all $\alpha \in L^n$.

Definition 9: A term, T_2, is said to properly contain another term, T_1, iff :

1. The index of T_2 is the same as the index of T_1.
2. T_1 subsumes T_2, and
3. for at least one $\alpha \in L^n$, $T_1(\alpha) < T_2(\alpha)$.

Note that a term subsumes itself but does not properly contain itself.

Definition 10: A function, f_2, includes a function, f_1, written $f_1 \leq f_2$, iff $f_1(\alpha) \leq f_2(\alpha)$ for all $\alpha \in L^n$. Conversely, f_1 is said to imply f_2 when $f_1 \leq f_2$. Note that a function both includes and implies itself.

Definition 11: An implicant, I of a function, f, is any product term that implies f.

Definition 12: A basic implicant is an implicant that contains no type 2 literals.

Definition 13: A basic prime implicant, BPI, of a function, f, is a basic implicant of f such that BPI $\leq$ BPI' implies that BPI is identical to BPI' for every basic implicant BPI' of f.

Definition 14: Let I_1 and I_2 be implicants of a function f, having indices K_1 and K_2 respectively. Then, I_1 is a primary implicant of f iff

1. I_1 is a basic prime implicant of f, i.e., $K_1 = 0$, or
2. For all I_2, either $K_1 > K_2$ or $I_1 \leq I_2$ implies that I_1 and I_2 having the same functional value.

Definition 15: Two product terms, $T_1 = r_1 \cdot X_1^{a_1} \cdot X_2^{a_2} \ldots X_n^{a_n}$ and $T_2 = 2.X_1^{b_1} \cdot X_2^{b_2} \ldots X_n^{b_n}$, are disjoint in coordinate i iff $X_i^{a_i} \cdot X_i^{b_i} = 0$ for all assignments of values to X_i. For example, $T_1 = 2 \cdot X_1^{002} X_2^{020} X_4^{200}$ and $T_2 = 1.X_1^{200} X_2^{220} X_3^{020} X_4^{x}$ are disjoint in coordinate 1 only.

Definition 16: The consensus in the i^{th} coordinate between two basic product terms $T_1 = r_1 \cdot X_1^{a_1} X_2^{a_2} \ldots X_n^{a_n}$ and $T_2 = r_2 \cdot X_1^{b_1} X_2^{b_2} \ldots X_n^{b_n}$ exists iff T_1 and T_2 are not disjoint in each coordinate $j \neq i$ and $X_i^{a_i} + X_i^{b_i} = X_i^{c_i}$ where a_i , $b_i \in (002,020,200,022,220,222)$ and $c_i \in (022,220,222)$.

Definition 17: When consensus in the i^{th} coordinate between two basic terms, T_1 and T_2, exists, it is given by:

$$T_3 = T_1 \overset{i}{*} T_2 = r_3 \cdot X_1^{c_1} \cdot X_2^{c_2} \ldots X_n^{c_n}, \text{ where}$$

$$r_3 = r_1 \cdot r_2$$

$$X_j^{c_j} = X_j^{a_j} \cdot X_j^{b_j} \text{ for } j \neq i, \text{ and}$$

$$X_i^{c_i} = X_i^{a_i} + X_i^{b_i}$$

Example: Consider consensus between

$$T_1 = 1 \cdot X_1^{220} X_2^{022} X_3^{022} \text{ and}$$

$$T_2 = 2 \cdot X_1^{022} X_2^{200} X_3^{220}$$

Neither $T_1 \overset{1}{*} T_2$ nor $T_1 \overset{3}{*} T_2$ exists because the 2^{nd} coordinate is disjoint. Since $r_3 = 1.2 = 1$, $X_1^{220} \cdot X_1^{022} = X_1^{020}$, $X_3^{022} \cdot X_3^{220} = X_3^{020}$, and $X_2^{022} + X_2^{200} = X_2^{222}$, T_3 is given by

$$T_3 = T_1 \overset{2}{*} T_2 = 1 \cdot X_1^{020} \cdot X_2^{222} \cdot X_3^{020}.$$

Definition 18: The join or the complement join in the i^{th} coordinate between two product terms

$$T_1 = r_1 \cdot X_1^{a_1} \cdot X_2^{a_2} \ldots X_n^{a_n} \text{ and}$$

$$T_2 = r_2 \cdot X_1^{b_1} \cdot X_2^{b_2} \ldots X_n^{b_n}$$

exists iff T_1 and T_2 are not disjoint in each coordinate $j \neq i$ and

(1) $r_1 \cdot X_i^{a_i} + r_2 \cdot X_i^{b_i} \geq (r_1 + r_2) \cdot X_i^{x}$ for join, or

(2) $r_1 \cdot X_i^{a_i} + r_2 \cdot X_i^{b_i} \geq (r_1 + r_2) \cdot X_i^{\bar{x}}$ for complement join.

Definition 19: When join or complement join in the i^{th} coordinate between product terms T_1 and T_2 exists, it is given by:

$$T_3 = T_1 \overset{i}{o} T_2 = r_3 \cdot X_1^{c_1} \cdot X_2^{c_2} \ldots X_n^{c_n} \text{ for join, or}$$

$$T_3 = T_1 \underset{i}{o} T_2 = r_3 \cdot X_i^{c_1} \cdot X_2^{c_2} \ldots X_n^{c_n} \text{ for complement join, where}$$

(1) $X_j^{a_j} \cdot X_j^{b_j} = S_j \cdot X_j^{c_j}$ for $j \neq i$, and

(2) $r_1 \cdot X_i^{a_i} + r_2 \cdot X_i^{b_i} \geq S_i \cdot X_i^{x} = (r_1 + r_2) \cdot X_i^{x}$ for join, or

$$r_1 \cdot X_i^{a_i} + r_2 \cdot X_i^{b_i} \geq S_i \cdot X_i^{\bar{x}} = (r_1 + r_2) \cdot X_i^{\bar{x}} \text{ for complement join, and}$$

(3) $r_3 = S_1 \cdot S_2 \ldots S_n$

The uniqueness of the expression $X_j^{a_j} \cdot X_j^{b_j} = S_j \cdot X_j^{c_j}$ is explained by Table 3. As an example, consider $X^a \cdot X^b = S \cdot X^c$ for a = 012, b = 022. From Table 2, we know that $X^{012} \cdot X^{022} = 2,1,0$ for X = 0,1,2 respectively and therefore the results S = 2 and c = 012 as in Table 3.

Example: Consider the join or complement join between

$$T_1 = 1.X_1^{022} \cdot X_2^{220} \ldots X_3^{220} \text{ and}$$

$$T_2 = 2.X_1^{200} \cdot X_2^{022} \ldots X_3^{x}$$

Join may be performed in first coordinate between T_1 and T_2 since they are not disjoint in all other coordinates and since

$$1.X^{022} + 2.X_1^{200} \geq (1+2).X_1^{x} = 2.X_1^{x}$$

Applying definition 19 and Table 3, the join is given by

$$T_3 = T_1 \overset{1}{o} T_2 = 2.2.2.X_1^{x} \cdot X_2^{020} \cdot X_3^{x} = 2.X_1^{x} \cdot X_2^{020} \cdot X_3^{x}.$$

The main reason for defining both join and complement join is that the resulting X and $\bar{X}$, or X^{012} and X^{210}, can be obtained directly from the non-binary memory device as proposed in [6] and, therefore, reduce the final cost.

Theorem 1: Every product term T containing type 2 literals has a functionally equivalent representation as a sum of basic product terms.

Theorem 2: Let T_1 and T_2 be two implicants of a function, f, then

(1) $T_1 \overset{i}{*} T_2$ and

(2) $T_1 \overset{i}{o} T_2$ and

(3) $T_1 \underset{i}{o} T_2$ are also implicants of f.

The proofs of Theorems 1 and 2 are contained inthe appendix. The interested reader is encouraged to refer to them. The proofs are not too difficult and will help the reader to gain a more thorough understanding of the algorithms that will be presented.

As a consequence of Theorem 2, terms resulting from the operation of consensus and join on implicants of f may be added to any representation of f without altering its functional values.

Definition 18: An input building block, IBB, is considered to be a basic logic operator. There are four distinct kinds of input building blocks, IBB1, IBB2, IBB3, and IBB4, which are defined as follows:

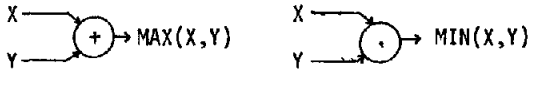
(1) $X \longrightarrow$ [IBB1] $\longrightarrow X^{002}$

(2) $X \longrightarrow$ [IBB2] $\longrightarrow X^{022}$

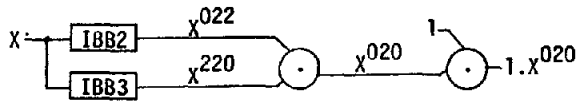
(3) $X \longrightarrow$ [IBB3] $\longrightarrow X^{220}$

(4) $X \longrightarrow$ [IBB4] $\longrightarrow X^{200}$

Definition 19: The MAX gate and the MIN gate are considered to be the logic operators which perform the operations of MAX and MIN respectively. They are defined as follows:



By using IBBs, X, $\bar{X}$, MAX and MIN gates, and the constants 1 and 2, one can implement a network to generate any desired function. For example, let $f(X) = 1.X^{020}$, then the implementation is as follows:



The process of generating the set of ternary prime implicants begins with the conversion of the given function, f, into a sum-of-basic-products form in which exactly one literal is present in each coordinate of every product term. Furthermore, these product terms must contain exactly one constant MIN'ed with the n literals. The operation of consensus is used repeatedly to generate all basic primary implicants. The operation of join is then used repeatedly to generate all primary implicants containin gone or more type 2 literals. All non-primary implicants are then dropped from the representation.

The last step of the simplification process is the procedure of covering, that is, the selection of a minimal-cost, static-and-dynamic-hazard-free representation from the list of primary implicants. The method we will be using is similar to the binary case McCluskey diagram [7] with added features to fulfil the requirements of minimal-cost and hazard-free. The basic requirement of a hazard free function is that for every allowable input transition, the pair of terms involved must be contained in at least one term in the final sum of product representation of the function. According to the definition of fundamental mode operation, each such pair of terms is actually a pair of terms which are adjacent to each other. A method has been developed to find all adjacent terms and will be described in section IV. Now let us consider the covering diagram. The diagram has a row for each prime implicant. The prime implicants are listed in increasing row order according to the sequence they have been generated. The diagram has a column corresponding to each $\alpha_j \varepsilon L^n$, for which $f(\alpha_j) > 0$, and also a column corresponding to each adjacent pair (α_j, α_k), where $\alpha_j \varepsilon L^n$ and $f(\alpha_j).f(\alpha_k) \neq 0$. A check is then placed in the i^{th} row and the j^{th} column if the prime implicant, PI_i, subsumes the term corresponding to the j^{th} column. After the covering diagram has been completed in this manner, the implicants near the bottom need fewer input building blocks than the ones near the top and also cost less. The reason is that higher-index implicants are generated from lower-index implicants and, therefore, are listed near the bottom of the covering diagram. By the definition of index, we know that the higher the index of a term is, the larger number of type 2 literals, i.e. X and $\bar{X}$, it contains. Therefore, fewer input building blocks are needed to form the term, hence, its cost is less. By applying the conventional covering technique from the bottom of the diagram to the top, we will generate the minimal-cost hazard-free representation of the function. Since the covering is done from the bottom of the diagram to the top, the terms near the bottom always have higher priority to be included in the final representation. It is important to point out that such a representation is unique due to this characteristic.

It should also be pointed out that as far as the cost is concerned only the factor of the number of input building blocks has been taken into account. If other factors, such as the speed of logic and the number of inputs etc., are also of concern, then at the final step of covering, a procedure other than the 'bottom to top' may be useful. For example, a 'top to bottom' procedure combined with the technique of row-dominance between terms with an identical constant may result in a representation that has certain desired properties. Further studies are needed to form a precise definition of a cost function.

IV. MINIMIZATION ALGORITHM AND IMPLEMENTATION

The minimization procedure consists of three algorithms. The entire procedure is specified by the following statements and, then, each algorithm is defined separately.

The minimization algorithm consists of 3 steps:

Step 1: Apply Algorithm 1 to find all basic (type 1) prime implicants of function f.

Step 2: Apply Algorithm 2 to find all type 2 prime implicants of function f.

Step 3: Apply Algorithm 3 to find the minimal-cost hazard-free representation of function f.

A computer program applying this algorithm has been implemented on a UNIVAC 1108/EXEC 8 at IIT. The program is written in FORTRAN. The data set is defined to handle up to 6 literals without modification and can be expanded to handle any number of literals with little change. The size of the program is about 3K words of instruction and 2K words of data and the CPU time for the example given in the next section is about 15 seconds.

These three algorithms are stated in the following paragraphs.

ALGORITHM 1: Apply the operation of consensus repeatedly to generate all basic (type 1) prime implicants of given function f.

Step 1.1: Take the consensus in all coordinates between all pairs of implicants in f. Don't care terms, if any, are considered as 2-terms. If a new term is produced which does not subsume any term present, add the new term to the end of f and then delete any subsuming terms. The consensus operation in all coordinates is repeated between all remaining terms and the newly added terms. Repeat this step until no new terms can be produced.

Step 1.2: The final result is then packed and renumbered starting from 0.

ALGORITHM 2: Apply the operations of join and complement join repeatedly to the result of Algorithm 1 to generate all type 1 and type 2 prime implicants of function f.

Step 2.1: Set K = 0. Let the type 1 prime implicants obtained from algorithm 1 be the starting set.

Step 2.2: Construct all possible joins and complement joins between terms of index K. Each term, T, resulting from the joins is examined as it is generated. If T has index K+1 and is not properly contained in another term of the same index already in the set, then T is added to the set. Otherwise, T is deleted.

Step 2.3: When a new term, T, of index K+1 is added to the set, all other terms of index K+1 are examined. Every such term properly contained in T is deleted from the set.

Step 2.4: If K+1 is less than the number of variables in f and at least one term of index K+1 has been added to the set, set K ← K+1 and go to Step 2.2.

ALGORITHM 3: Apply the following covering technique to choose a minimal-cost, hazard-free representation for the given operation hazard free function f.

Step 3.1: Set up rows for the covering table. The primary implicants generated from algorithm 2 are listed as rows according to their generated order.

Step 3.2: Set up columns for the covering table. First, there is a column for each 1-term and 2-term (but not for don't care terms) of the input to algorithm 1. Secondly, there is a column corresponding to each pair of adjacent terms. The adjacent terms are generated by the following method.

N = number of variables in the function

E = the entry ID number (refer to Table 4 for example) for the term being considered

T = an array which stores the corresponding functional values of terms indexed by their ID's.

(A(J),J=1,n) = E's adjacent entries

```
begin J ← 0;
  for i ← 0 step 1 until N-1 do
    begin if T(E)=0 then no-adjacent-
            terms;
          if (E-3^(i+1).(E/3^(i+1))≥2*3^i) then
            goto break;
          if (E+3^i>3^n-1) then goto
            break:
          CANDIDATE ← E+3^i;
          if T(CANDIDATE)=0 then
            goto break;
          J ← J + 1;
          A(J) ← E + 3^i
    break: end
  end
```

Step 3.3: A check is placed in the i^{th} row and the j^{th} column if the prime implicant, PLi, subsumes the term corresponding to the j^{th} column.

Step 3.4: The completed covering table is inspected for columns containing only a single check. The primary implicants (rows) that cover a single check in their column are called essential primary implicants. A check mark is placed in the table next to the essential primary implicants to indicate that they have been selected.

Step 3.5: Check each column which is covered by the essential primary implicants.

Step 3.6: For those columns which are not checked in step 3.5, apply step 3.7 repreatedly until all columns have been checked. Go to step 3.8 if all columns have been checked.

Step 3.7: Start from the bottom of the table and go upward, selecting the first primary implicant (row) which does not have a check placed in front of it. A check mark is placed in the table next to this selected term and then check each column which is covered by this term.

Step 3.8: The set of all the primary implicants (rows) with a check mark placed in front forms the final sum-of-products representation for the function. It is important to point out that this process generates a unique final representation.

V. FINAL EXAMPLE

The function used to demonstrate the minimization procedure is chosen based on the fact that both the join and the complement join operations are applied, and the minimization procedure is non-trivial as far as hand implementation is concerned.

Consider the function, f, whose tabular representation is given in Table 4. The E-number at the lower right corner of each entry is used to uniquely identify the entry. The input to algorithm 1 is listed in Table 5. Since there are three distinct variables, X_1, X_2, and X_3, in this example, a total of $3^3 = 27$ input terms are expected. The notation used in Tables 5 - 7 is equivalent to the data base used for the computer implementation. For example, item no. 1 in Table 5 indicates the input term is $1 \cdot X_1^{002} X_2^{002} X_3^{020}$. Output from algorithm 1, which is also the input to algorithm 2, is listed in Table 6. There are 11 basic prime implicants generated at this stage. Instead of a step by step explanation for the entire procedure, a single generated term is traced to demonstrate the operation. For example, item no. 2 in Table 6 is identical to item no. 18 in Table 5 because no consensus operation can be applied on any of the literals between item no. 18 in Table 5 and any other terms in Table 5.

Output from algorithm 2, which is also the input to algorithm 3, is listed in Table 7. There are a total of 48 type-2 prime implicants including 27 index-1 terms, 16 index-2 terms, and 4 index-3 terms. Again, only a single output term is traced to demonstrate the operation. For example, item no. 11 in Table 7 is generated by applying the operation of join on the third literal between items 0 and 7 of Table 6.

The last step of the minimization process is the process of covering. First of all, adjacent terms of the function are located. The result is listed in Table 8. The last column in this table indicates the two adjacent entries as identified in Table 4. For example, item no. 1 indicates that there exists a pair of adjacent terms (E1,E2), which are $X_1^{002}\ X_2^{002}\ X_3^{020}$ and $X_1^{002}\ X_2^{002}\ X_3^{200}$ respectively, with a common constant 1. There are a total of 42 adjacent pairs in this function. Now we can build up a covering table with a row for each item in Table 7 and a column for each nonzero entry in Table 4 and also a column for each adjacent pairs in Table 8. To make the understanding of the covering procedure easier, Table 9 provides the tabular representations for items 55, 56, 57, and 58 of Table 7. A partial covering diagram is then shown in Table 10. It is important to recall that the covering is done from the bottom of the diagram up to the top. Only the essential part of the diagram is shown. From this diagram, we know that the final representation is

$$f(X)=2.X_1^{012}.X_2^{012}.X_3^{210}+2.X_1^{012}.X_2^{210}.X_3^{012} + 2.X_1^{210}X_2^{012}X_3^{012}+2.X_1^{210}X_2^{210}X_3^{210}$$
$$=2.\overline{X}_1.\overline{X}_2.X_3+2.\overline{X}_1.X_2.\overline{X}_3+2.X_1.\overline{X}_2.\overline{X}_3+ 2.X_1.X_2.X_3 \quad (5)$$
$$\qquad A \qquad\qquad B \qquad\qquad C \qquad\qquad D$$

Now we will informally prove that (5) is indeed the expected result. The proof contains two parts. Firstly we shall prove that (5) is hazard free. Refer to Table 4 for entry names to be used in the following analysis. Let E13 be the initial state of the network. According to the fundamental mode of asynchronous operation, the possible next state is an element in the set (E4, E10, E12, E14, E16, E22). A hazard free network requires that the entries involved in a possible transition be included in at least one term. This is exactly the case in (5), namely (E13, E4) is in terms A and B, (E13, E10) is in terms B and C, (E13, E12) is in terms A and C, (E13, E14) is in terms B and D, (E13, E16) is in terms A and D, and (E13, E22) is in terms C and D. A similar approach can be applied to prove that all possible transition for this function (5) is hazard free. Secondly, we shall prove that the cost of this representation is minimum. Since (5) contains no input building blocks at all, its cost must be the minimum. From the above informal proof, we conclude that (5) is the minimal-cost hazard-free representation for the OHF function f.

As a contrast example, the above covering procedure is repeated on the same diagram. Only this time, it is done from the top of the diagram down to the bottom. The result is shown on Table 10 and is as follows:

$$f(X)=2.X_1^{002}X_2^{002}X_3^{200}+2.X_1^{002}X_2^{200}X_3^{002} + 2.X_1^{200}X_2^{002}X_3^{002} + 2.X_1^{200}X_2^{200}X_3^{200} + 1.X_1^{022}X_2^{220}X_3^{022} + 1.X_1^{220}X_2^{022}X_3^{022} + 1.X_1^{220}X_2^{220}X_3^{220} + 1.X_1^{222}X_2^{222}X_3^{020} + 1.X_1^{222}X_2^{020}X_3^{222} + 1.X_1^{022}X_2^{022}X_3^{220}$$

which is much more expensive than (5).

VI. CONCLUSION

A minimization algorithm for ternary switching functions has been described. Further studies are needed to form a precise definition of a cost function, namely the number of inputs, the number of grates, the number of input building blocks, and speed of the logic and the relationships among them should all be considered as cost factors to determine the minimal-cost representation of a function.

The definition of operation-hazard-free function can be somewhat extended because even if relation (1) in section I does not hold for any pair of adjacent entries, a network free of static and dynamic hazards could still be designed provided that there is no direct transfer, based on the corresponding state flow diagram, from one entry to the other. An algebraic description and an algorithm to incorporate this characteristic into the minimization process is under investigation.

A simulator is being implemented to simulate the functional characteristics of ternary switching network. The input to this simulator is the result of this minimization process, and the output will be applied to other research areas such as the study of speed-independent asynchronous ternary networks.

VII. REFERENCES

[1] Post, E.L., "Introduction To a General Theory of Elementary Propositions", American Jour. of Math., vol. 43, 1921, pp. 163-185.

[2] Wojcik, A.S., "Multi-Valued Asynchronous Circuits", Conference Record of the 1973 International Symposium on Multiple-Valued Logic, Toronto, Canada, May 1973.

[3] Yoeli, M., and C. Rosenfeld, "Logical Design of Ternary Switching Circuits", IEEE Transactions on Electronic Computers, vol. EC14, Feb. 1965, pp. 19-29.

[4] Quine, W.V., "On Cores and Prime Implicants of Truth Function", Am. Math. Monthly, vol. 66, Nov. 1959, pp. 755-760.

[5] Allen, C.M., "Comments on Ternary Switching Circuits", An unpublished paper, State University of New York at Buffalo, Dec. 1973.

[6] Wojcik, A.S., "Multi-Valued Asynchronous Sequential Circuits", Conference Record of the 1974 International Symposium on Multi-Valued Logic, Morgantown, West Virginia, May 1974.

[7] McCluskey, E.J., "Introduction to the Theory of Switching Circuits", McGraw-Hill, 1965.

APPENDIX

Proof of Theorem 1:

Since $X_i^{x} = 1.X_i^{020} + 2.X_i^{200}$ and $X_i^{\bar{x}} = 2.X_i^{002} + 1.X_i^{020}$, each occurence of X_i^{x} or $X_i^{\bar{x}}$ in the product term, T, may be replaced by its equivalent, $(1.X_i^{020} + 2.X_i^{200})$ or $(2.X_i^{002} + 1.X_i^{020})$ respectively. Furthermore, since '.' is distributive over '+' the resulting expression may be expanded into a sum of product terms containing no type 2 literals.

Proof of Theorem 2:

(1) Let $T_1 = r_1.X_1^{a_1}.X_2^{a_2} \ldots X_n^{a_n}$ and

$T_2 = r_2.X_1^{b_1}.X_2^{b_2} \ldots X_n^{b_n}$.

Since T_1 and T_2 are two implicants of the function, f, it follows that $T_1 \leq f$, $T_2 \leq f$, and $T_1 + T_2 \leq f$.

Let $T_1' = X_1^{a_1}.X_2^{a_2} \ldots X_{i-1}^{a_{i+1}} \ldots X_n^{a_n}$ and

$T_2' = X_i^{b_1}.X_2^{b_2} \ldots X_{i-1}^{b_{i+1}} \ldots X_n^{b_n}$.

By definition, we know that

$T_1 = r_1.X_i^{a_i}.T_1'$ and $T_2 = r_2.X_i^{b_i}.T_2'$.

Therefore, $T_1 + T_2 \leq f$ can be rewritten as

$r_1.X_i^{a_i}.T_1 + r_2.X_i^{b_i}.T_2' \leq f$.

Since $T_1.T_2' \leq T_1+T_2$ and $T_1'.T_2 \leq T_1+T_2$ it follows that

$T_1.T_2' + T_1'.T_2 \leq T_1 + T_2$,

or $r_1.X_i^{a_i}.T_1'.T_2' + r_2.X_i^{b_i}.T_1'.T_2' \leq T_1+T_2$,

or $(r_1.X_i^{a_i} + r_2.X_i^{b_i}).T_1'.T_2' \leq T_1+T_2$. (A1)

Since $r_1.R_2(r_1.X_i^{a_i} + r_1.R_2.X_i^{b_i}).T_1'.T_2' \leq (r_1.X_i^{a_i} + r_2.X_i^{b_i}).T_1'.T_2'$,

it follows that

$(r_1.r_2.X_i^{a_i} + r_1.r_2.X_i^{b_i}).T_1'.T_2' \leq T_1+T_2$,

or $(X_i^{a_i} + X_i^{b_i}).r_1.r_2.T_1'.T_2' \leq T_1+T_2$. (A2)

By definition, the left hand side of the final inequality (A2) is $T_1 \overset{i}{*} T_2$ (when consensus in the i^{th} coordinate between T_1 and T_2 exists). Hence,

$T_1 \overset{i}{*} T_2 \leq T_{1_i} + T_2$

which implies $T_1 * T_2$ is an implicant of f.

(2) By definition, the left hand side of the established inequality (A1) is $T_1 \overset{i}{o} T_2$ (when join in the i^{th} coordinate between T_1 and T_2 exists). Hence,

$T_1 \overset{i}{o} T_2 \leq T_{1_i} + T_2$

which implies $T_1 \overset{i}{o} T_2$ is an implicant of f.

(3) By definition, the left hand side of the established inequality (A2) is $T_1 \overset{i}{\underline{o}} T_2$ (when complement join in the i^{th} coordinate between T_1 and T_2 exists). Hence,

$T_1 \overset{i}{\underline{o}} T_2 \leq T_1 + T_{2_i}$

which implies $T_1 \underline{o} T_2$ is an implicant of f.

Table 1 - A simple OHF function

X_1 \ X_2	0	1	2
0	0_A	1_B	2_C
1	0_D	1_E	1_F
2	0_G	0_H	0_I

Table 2 - A Selected Set of Unary Functions

X \ f(X)	X^{210}	X^{012}	X^{002}	X^{020}	X^{022}	X^{220}	X^{200}	X^{222}
				(X^{x})	($X^{\bar{x}}$)			
0	0	2	2	0	2	0	0	2
1	1	1	0	2	2	2	0	2
2	2	0	0	0	0	2	2	2

Table 3 - $X^a \cdot X^b = S \cdot X^c$

a \ b	002	012	020	022	200	210	220	222
002	$2/_{002}$	$2/_{002}$	D	$2/_{002}$	D	D	D	$2/_{002}$
012	$2/_{002}$	$2/_{012}$	$1/_{020}$	$2/_{012}$	D	$1/_{020}$	$1/_{020}$	$2/_{012}$
020	D	$1/_{020}$	$2/_{020}$	$2/_{020}$	D	$1/_{020}$	$2/_{020}$	$2/_{020}$
022	$2/_{002}$	$2/_{012}$	$2/_{020}$	$2/_{022}$	D	$1/_{020}$	$2/_{020}$	$2/_{022}$
200	D	D	D	D	$2/_{200}$	$2/_{200}$	$2/_{200}$	$2/_{200}$
210	D	$1/_{020}$	$1/_{020}$	$1/_{020}$	$2/_{200}$	$2/_{210}$	$2/_{210}$	$2/_{210}$
220	D	$1/_{020}$	$2/_{020}$	$2/_{020}$	$2/_{200}$	$2/_{210}$	$2/_{220}$	$2/_{220}$
222	$2/_{002}$	$2/_{012}$	$2/_{020}$	$2/_{022}$	$2/_{200}$	$2/_{210}$	$2/_{220}$	$2/_{222}$

notes: 1. each entry is S/C
2. D indicates disjoint relationship

Table 4 - A Final Example

X_1 \ X_2X_3	00	01	02	10	11	12	20	21	22
0	0_{E0}	1_{E1}	2_{E2}	1_{E3}	1_{E4}	1_{E5}	2_{E6}	1_{E7}	0_{E8}
1	1_{E9}	1_{E10}	1_{E11}	1_{E12}	1_{E13}	1_{E14}	1_{E15}	1_{E16}	1_{E17}
2	2_{E18}	1_{E19}	0_{E20}	1_{E21}	1_{E22}	1_{E23}	0_{E24}	1_{E25}	2_{E26}

Note: The E-number at the lower right corner in each cell is the entry ID number.

Table 5 - Input to algorithm 1

item no.	constant r	literal 1	literal 2	literal 3
0	0	002	002	002
1	1	002	002	020
2	2	002	002	200
3	1	002	020	002
4	1	002	020	020
5	1	002	020	200
6	2	002	200	002
7	1	002	200	020
8	0	002	200	200
9	1	020	002	002
10	1	020	002	020
11	1	020	002	200
12	1	020	020	002
13	1	020	020	020
14	1	020	020	200
15	1	020	200	002
16	1	020	200	020
17	1	020	200	200
18	2	200	002	002
19	1	200	002	020
20	0	200	002	200
21	1	200	020	002
22	1	200	020	020
23	1	200	020	200
24	0	200	200	002
25	1	200	200	020
26	2	200	200	200

Table 6 - Output from algorithm 1 or input to algorithm 2

item no.	constant r	literal 1	literal 2	literal 3
0	2	002	002	200
1	2	002	200	002
2	2	200	002	002
3	2	200	200	200
4	1	022	220	022
5	1	220	022	022
6	1	220	220	220
7	1	222	222	020
8	1	222	020	222
9	1	022	022	220
10	1	020	222	222

Table 7 - Output from algorithm 2

item no.	constant r	literal 1	literal 2	literal 3	type	index
0	2	002	002	200	1	0
1	2	002	200	002	1	0
2	2	200	002	002	1	0
3	2	200	200	200	1	0
4	1	022	220	022	1	0
5	1	220	022	022	1	0
6	1	220	220	220	1	0
7	1	222	222	020	1	0
8	1	222	020	222	1	0
9	1	022	022	220	1	0
10	1	020	222	222	1	0
11	2	002	002	210	2	1
12	2	002	012	200	2	1
13	2	012	002	200	2	1
14	2	012	200	002	2	1
15	2	002	210	002	2	1
16	2	002	200	012	2	1
17	2	210	002	002	2	1
18	2	200	012	002	2	1
19	2	200	002	012	2	1
20	2	210	200	200	2	1
21	2	200	210	200	2	1
22	2	200	200	210	2	1
23	1	012	220	022	2	1
24	1	210	022	022	2	1
25	1	210	220	220	2	1
26	1	210	222	020	2	1
27	1	012	222	020	2	1
28	1	222	210	020	2	1
29	1	222	012	020	2	1
30	1	222	020	210	2	1
31	1	222	020	012	2	1
32	1	022	022	210	2	1
33	1	020	222	210	2	1
34	1	020	222	012	2	1
35	1	210	020	222	2	1
36	1	012	020	222	2	1
37	1	020	210	222	2	1
38	1	020	012	222	2	1
39	2	002	012	210	2	2
40	2	012	002	210	2	2
41	2	012	012	200	2	2
42	2	012	210	002	2	2
43	2	012	200	012	2	2
44	2	002	210	012	2	2
45	2	210	012	002	2	2
46	2	210	002	012	2	2
47	2	200	012	012	2	2
48	2	210	200	210	2	2
49	2	210	210	200	2	2
50	2	200	210	210	2	2
51	1	012	210	022	2	2
52	1	210	012	022	2	2
53	1	210	220	210	2	2
54	1	022	012	210	2	2
55	2	012	012	210	2	3
56	2	012	210	012	2	3
57	2	210	012	012	2	3
58	2	210	210	210	2	3

Table 8 - Adjacent Terms of Final Example

item no.	constant r	entries included (refer to Table 4)
1	1	(E1,E2)
2	1	(E1,E4)
3	1	(E1,E10)
4	1	(E2,E5)
5	1	(E2,E11)
6	1	(E3,E4)
7	1	(E3,E6)
8	1	(E3,E12)
9	1	(E4,E5)
10	1	(E4,E7)
11	1	(E4,E13)
12	1	(E5,E14)
13	1	(E6,E7)
14	1	(E6,E15)
15	1	(E7,E16)
16	1	(E9,E10)
17	1	(E9,E12)
18	1	(E9,E18)
19	1	(E10,E11)
20	1	(E10,E13)
21	1	(E10,E19)
22	1	(E11,E14)
23	1	(E12,E13)
24	1	(E12,E15)
25	1	(E12,E21)
26	1	(E13,E14)
27	1	(E13,E16)
28	1	(E13,E22)
29	1	(E14,E17)
30	1	(E14,E23)
31	1	(E15,E16)
32	1	(E16,E17)
33	1	(E16,E25)
34	1	(E17,E26)
35	1	(E18,E19)
36	1	(E18,E21)
37	1	(E19,E22)
38	1	(E21,E22)
39	1	(E22,E23)
40	1	(E22,E25)
41	1	(E23,E26)
42	1	(E25,E26)

Table 9 - Tabular representation for terms 55 thru 58

$55 - 2.X_1^{012}\ X_2^{012}\ X_3^{210}$ - A

X_1 \ X_2X_3	00	01	02	10	11	12	20	21	22
0	0	0	0	1_{E3}	1_{E4}	0	2_{E6}	1_{E7}	0
1	0	0	0	1_{E12}	1_{E13}	0	1_{E15}	1_{E16}	0
2	0	0	0	0	0	0	0	0	0

$56 - 2.X_1^{012}\ X_2^{210}\ X_3^{012}$ - B

X_1 \ X_2X_3	00	01	02	10	11	12	20	21	22
0	0	1_{E1}	1_{E2}	0	1_{E4}	1_{E5}	0	0	0
1	0	1_{E10}	1_{E11}	0	1_{E13}	1_{E14}	0	0	0
2	0	0	0	0	0	0	0	0	0

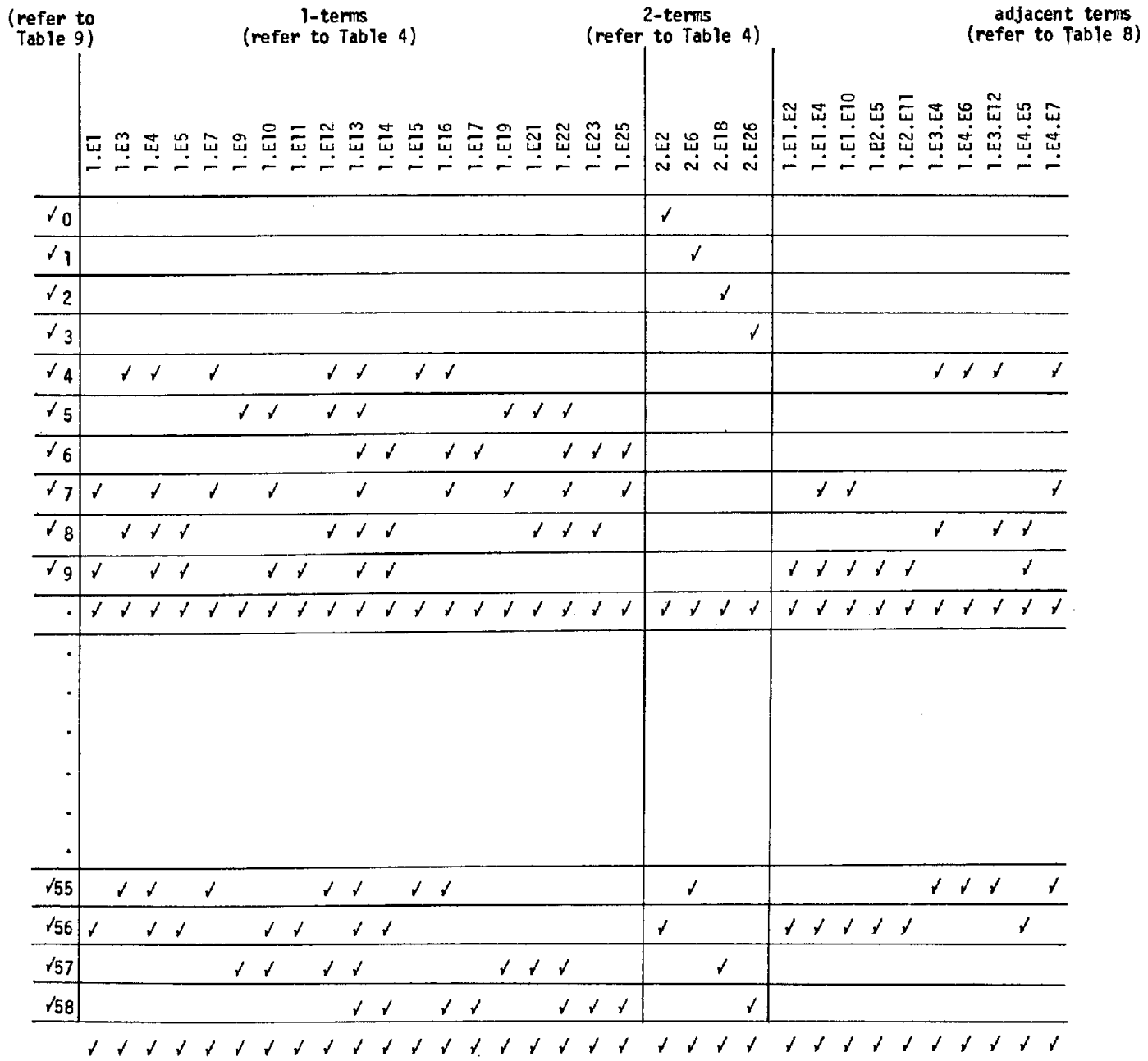
$57 - 2.X_1^{210}\ X_2^{012}\ X_3^{012}$ - C

X_1 \ X_2X_3	00	01	02	10	11	12	20	21	22
0	0	0	0	0	0	0	0	0	0
1	1_{E9}	1_{E10}	0	1_{E12}	1_{E13}	0	0	0	0
2	2_{E18}	2_{E19}	0	1_{E21}	1_{E22}	0	0	0	0

$58 - 2.X_1^{210}\ X_2^{210}\ X_3^{210}$ - D

X_1 \ X_2X_3	00	01	02	10	11	12	20	21	22
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1_{E13}	1_{E14}	0	1_{E16}	1_{E17}
2	0	0	0	0	1_{E22}	1_{E23}	0	1_{E25}	1_{E26}

Table 10 - Covering Diagram



58
57
56
55
9
8
7
6
5
4
3
2
1
0
1.E4.E13
1.E5.E14
1.E6.E7
1.E6.E15
1.E7.E16
1.E9.E10
1.E9.E12
1.E9.E18
1.E10.E11
1.E10.E13
1.E10.E19
1.E11.E14
1.E12.E13
1.E12.E15
1.E12.E21
1.E13.E14
1.E13.E16
1.E13.E22
1.E14.E17
1.E14.E23
1.E15.E16
1.E16.E17
1.E16.E25
1.E17.E26
1.E18.E19
1.E18.E21
1.E19.E22
1.E21.E22
1.E22.E23
1.E22.E25
1.E23.E26
1.E25.E26