論文81－18－1－4

Fault Analysis in Multivalued Combinational Circuits Using the Boolean Difference Concept

(부울 微分을 이용한 多値 論理 回路에서의 缺陷 解析)

柳光烈*, 金宗相**

(Ryu, Kwang Ryel and Kim, Chong Sang)

要　約

부울 微分의 개념을 응용하여, Allen-Givone implementation oriented algebra에 의한 多値 논리 회로내의 缺陷을 해석했다. 회로내의 모든 라인을 그 성질에 따라 다섯 가지 유형으로 분류하였으며 각 유형별로 완전한 테스트 세트를 표현하는 식을 유도하고 증명했다. 이들 식의 실제 응용예의 결과는 진리표와의 비교에 따라 옳음이 확인되었다.

Abstract

Any logical stuck-at faults in multivalued combinational circuits are analyzed using the concept of Boolean difference. The algebra employed is the implementation oriented algebra developed by Allen and Givone. All the lines in the circuit are classified into five types according to their properties. For each type, the equation that represents the complete test set is derived and proved. All the results in examples are confirmed to be correct by comparing the truth tables of the normal and faulty circuits.

I. Introduction

Recently, fault analysis in binary system has been one of the principal research areas in digital systems. For the multivalued logic, little work has been reported in fault detection and location area, although a number of multivalued algebras are developed.

One of the main advantages of multivalued logic is that the required number of pins for an integrated circuit chip to communicate with its outside world can be decreased. In addition, one can often avoid the cost of coding multivalued information into binary form for processing and then decoding it back to the multivalued signal after the manipulation is finished. A computer which uses multivalued arithmatic can be consideribly faster because it takes fewer digits to represent a decimal number in the multivalued number system compared to the binary case. In 1958 the first full scale implementation of a ternary computer, SETUN, was completed in Russia. A computer emulation, TERNAC, was implemented in 1973 at the State University of New York, Amherst, New York.

However, multivalued circuits are not in wide use because the cost of building such circuits is relatively high while the reliability is not as high as might be desired. Methods of ensuring reliability and procedure for testing the circuits must be developed.

R.J. Spillman and S. Y. H. Su developed a modified D-algorithm to detect logical stuck-at fault in multivalued logic circuits.[1] The algebra employed was the generalized ternary algebra developed by Vranesic, Lee, and Smith.[2] Their paper indicates that the generalized ternary algebra simplifies fault detection, however, it may complicate fault detection.

This paper is concerned with the analysis of logical stuck-at faults in multivalued combinational

* 準會員, 서울大 大學院 電子工學科
(Dept. of Electronics Engineering, Seoul National University)

** 正會員, 서울大 電子計算機工學科
(Dept. of Computer Engineering, Seoul National University)

circuits based on the Allen-Givone implementation. The Boolean difference technique is used to construct equations which represents the complete test set for the circuit. Section II introduces the Allen-Givone implementation oriented algebra and the minimization methods of multivalued switching functions. In section III, we examine the Boolean difference concept and its extention for application in multivalued circuits. All the lines in the circuit are classified into five types and the equations of the complete test set are derived for each type.

II. The Allen – Givone Implementation Oriented Algebra

The fundamental principle underlying multiple-valued logic systems is that they have n input variables and t output variables such that there are at most m values for each of the physical variables, where the physical values are $v_1, v_2, \ldots, v_m$. Furthermore, if $v_1<V_2<\ldots<Vm$, the integer 0 can be assigned to v_1, 1 to v_2, ... , m-1=p to v_m. Then, each input and output variable may assume at any instant one of the set of m logic values from L, where L = {0,1,2,... ,p} and p=m-1.

It can easily be shown that Boolean algebras cannot be used as adequate mathematical models for multiple-valued logic systems, since even for a simple three-valued system, no Boolean algebras exist. The algebra here is an implementable multiple-valued switching algebra. Two operations can now be defined,[3] (+) and (•), by

$$x + y = \max(x, y) \tag{1.a}$$

$$x \cdot y = \min(x, y) \tag{1.b}$$

where $x, y \in L = \{0, 1, 2, \ldots, p\}$. The unary operator (a, b) on the variable x, called a literal and denoted by $\overset{ab}{x}$ is defined by

$$\overset{ab}{x} = \begin{cases} p, \text{ if } a \leq \text{logical value of } x \leq b \\ 0, \text{ otherwise.} \end{cases} \tag{2}$$

where $a, b \in L$ and $a \leq b$.
And the complement of $\overset{ab}{x}$, denoted by $\overset{\overline{ab}}{x}$, is defined by

$$\overset{\overline{ab}}{x} = \begin{cases} o, \text{ if } a \leq \text{logical value of } x \leq b \\ p, \text{ otherwise.} \end{cases} \tag{3}$$

By mapping the multivalued variable x to a binary variable $\overset{ab}{x}$ and introducing the definition of complement of $\overset{ab}{x}$, a multivalued switching algebra has been formed which satisfies all axioms and theorems of Boolean algebra and can be used for minimization of any partially specified multivalued switching functions in a similar way as the minimization of Boolean functions.

Minimization

Any multi-valued switching functions and their complements can be minimized in terms of the number of literals or number of gates for a two-level logic networks. For function f with a small number of variables n (say, $n\leq4$) and small p (say, $p\leq4$), f and $\overline{f}$ can be minimized either by hand calculation using theorems of multivalued algebra or by map method.[4-6] For functions of larger n and p, cubid notations are used and computer algorithms for minimizing any multivalued switching functions are presented. Here, only the map minimization method will be illustrated in Example 1.

Generally, for a given (p+1)-valued function of n variables, we can express f as[5]

$$f = 1\cdot f_1 + 2\cdot f_2 + \ldots + p\cdot f_p + f_{dc} \tag{4}$$

$$= \left(\sum_{i=1}^{p} i \cdot f_i\right) + f_{dc}$$

where

$$f_i = \sum_{j=1}^{q_j} P_j \tag{5}$$

and P_j is corresponding to the input n tuple for which f = logic i, P_j is the product $\Pi \overset{ai\,bi}{x_j}$. q_j is the number of input n tuples for which f = logic i.

$$f_{dc} = \sum_{k=1}^{q_d} d_k P_k \tag{6}$$

where P_k is a product term corresponding to the input n tuple for which f = DON'T CARE. d_k can be assigned to any value in the set 0,1,2,...p and q_d is the number of input n tuples for which f = DON'T CARE.

To minimize f we sequentially minimize f_i starting with i = p followed by i = p-1, p-2, ..., 1, taking the DON'T CAREs into consideration. As far as f_i is concerned, all input n tuples for which f = j (i.e. product terms of f_j) where j>i can be treated as DON'T CAREs and they can be assigned to logic i if it is advantageous to do so.[5] This idea will be used in the following example.

Example 1; The map representation of a multiple-valued switching function is essentially just a rearrangement of its table of combinations. A map for a two – variable three – valued switching function is given in Fig. 1.

First, all rectangular groupings of cells containing the value 2 are found. Next all rectangular groupings containing a logic 1 or higher (not totally contained in a larger grouping) are shown. Thus, there are four prime implicants of this function, and it can be seen that these prime implicants[4] form a minimal representation of f.

$$f = 2 \cdot ({}^{0}x^{0}_{1} \cdot {}^{0}x^{0}_{2} + {}^{2}x^{2}_{1} \cdot {}^{1}x^{2}_{2})$$
$$+ 1 \cdot ({}^{0}x^{0}_{1} + {}^{0}x^{1}_{1} \cdot {}^{0}x^{0}_{2}) \qquad (7)$$

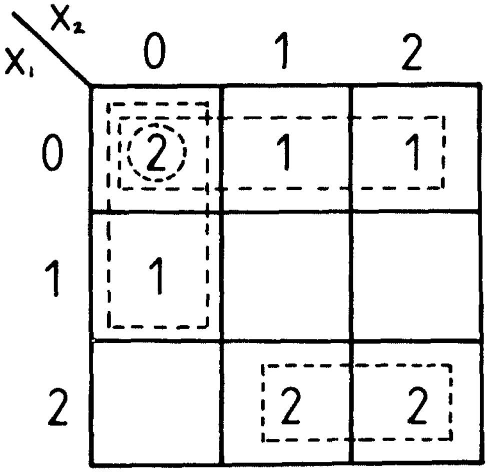

Fig. 1. A map for a two-variable three-valued switching function.

This is three – valued function and 2 is the maximum value, so the number 2 before the first parentheses may be omitted in the above expression. The circuit realization of this function is shown in Fig. 2.

III. Fault Analysis

Fault Model

In this paper, a fault in any gates will be defined as an input or an output line stuct – at a given logic

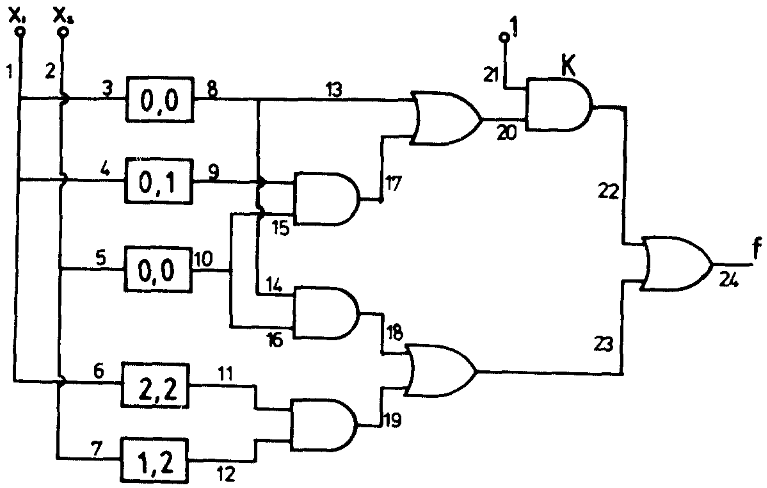

Fig. 2. $f = ({}^{0}x^{0}_{1}\,{}^{0}x^{0}_{2} + {}^{0}x^{0}_{2} + {}^{2}x^{2}_{1}\,{}^{1}x^{2}_{2}) + 1 \cdot ({}^{0}x^{0}_{1} + {}^{0}x^{1}_{1}\,{}^{0}x^{0}_{2})$.

value, i.e., s-a-k. The circuit is assumed to be irredundant and only a single fault can be present at any time. Certain different gate level faults produce the same output under all possible inputs. Such faults can be detected as a group, but the individual faults within each group cannot be distinguished. These faults are called indistinguishable faults. Specifically, two faults, i and j, and indistinguishable if for all possible inputs, the output with failure i is the same as the output with failure j. The indistinguishable faults for the binary OR gate, for example, are any input stuck – at -1 and any output stuck – at -1. The indistinguishable faults for the multivalued gates will be discussed on each case in the following description of fault analysis.

Basci Concept

We use the concept of the Boolean Difference to analyze any single stuck type faults in multivalued combinational circuits.

The Boolean Difference of a binary function $f(x_1, x_2, ..., X_n)$ with respect to the variable x_i, denoted by $df(x_1, x_2, ..., x_n)/d_{x_i}$ is defined as follows;[7-8]

$$\begin{aligned}\frac{df(X)}{dx_i} &= f\,(x_1, x_2, ...x_i, ..., x_n) \\ &\quad \oplus f\,(x_1, ...\overline{x}_i, ..., x_n) \\ &= f\,(x_1, ..., 0, ..., x_n) \\ &\quad \oplus f\,(x_1, ...1, ..., x_n)\end{aligned} \tag{8}$$

In the binary circuits, it is well known that the necessary and sufficient conditions on the Boolean variables $x_1, ..., x_n$, that is, the primary inputs such that the output of the circuit realizing $f(x_1, ..., x_n)$ is dependent upon the logical value of line j, x_j, is that

$$\frac{df\,(x_1, ..., x_n, x_j)}{dx_j} = 1 \tag{9}$$

To test for the specific fault x_j s-a-a_k, $a_k \in$ 0, 1 , we must make the output of the circuit dependent upon x_j, and drive line j to the logic value $\overline{a}_k$. These conditions are met if and only if[9-10]

$$x_j^{\overline{a}k} \cdot \frac{df\,(x_1, ..., x_n, x_j)}{dx_j} = 1 \tag{10}$$

where

$$x_j^{\overline{a}k} = \begin{cases} x_j, \text{ if } a_k = 0 \\ \overline{x}_j, \text{ if } a_k = 1 \end{cases}$$

Now, consider the method of finding out the complete test set for single stuck type faults in multivalued combinational logic circuits based on the implementation oriented algebra developed by Allen and Givone.

As mentioned above, in this algebra, the multivalued variables are mapped to the binary variables by introducing ${}^{a}x^{b}$ gates, thus it is possible to analyze faults by using the concept of Boolean difference. But, there are some difficult problems. First, not all the lines in the circuit have only the binary values (o, p). The input lines of ${}^{a}x^{b}$ gates and the output lines of the last AND gates of each "Group" (for example, gate K in Fig. 2.) can take on any intermediate logical values between o and p. Here, the Group means the circuit module which realizes a subfunction of f, i.e. i.f_i in E_q.(4) And k – Group means the circuit module of k•f_k. Another problem is that, when minimizing f_i, all input n tuples for which f = j where j > i can be treated as DON'T CAREs. For these reasons, the method of deriving tests for a particular fault becomes more complicated in multivalued circuits than in binary circuits.

Taking into account all these properties and extenting the concept of equation (10), the set of tests in multivalued case for the stuck – at – a_i fault on line i which has the binary value in k – Group is represented by the following basic form of equation;

$$X_i(a_i) \cdot \frac{dF_k}{dX_i} \cdot \prod_j \overline{f}_j = p \tag{11}$$

where F_k is the representation of f_k as a function of faulty line X_i, and $X_i(a_i)$ is X_i or $\overline{X}_i$ depending on the value of a_i, and dF_k/dX_i is defined by equation(8) except that logic 1 is mapped to logic p. In other words, $X_i(a_i) = p$ is the condition that line i is driven to the logic value $\bar{a}_i$, while $dF_k/dX_i = p_{is}$ the necessary and sufficient condition that the output of k – Group of the circuit is dependent upon the logical value of line i, X_j. since the Boolean difference is applied only to k – Group, the effect of the specific fault in k – Group to the other Group as well as the effect of DON'T CAREs should be considered in deriving the complete test set for that fault. By adding the last term, $\prod_j \bar{f}_j = p$, all these conditions are met. Equation (11) is sometiomes modified and completely determined depending upon the properties of faulty line. For the faults on the other lines which have non – binary values, tests can be derived by another equations. Boolean difference cannot be applied to these lines, but the form of these equations is similar to equation (11) except the Boolean difference term.

Line classification

It is very important in fault analysis to classify all the lines in the circuit by their properties. There are five types of lines.

Type 1: The input lines of ${}^{a}x^{b}$ gates (for example, 1, 2, 3, 4, 5, 6, 7, in Fig. 2)

Type 2: The output line (24)

Type 3: The input lines of the last OR gate (22, 23)

Type 4: The enabling input lines of the last AND gate of each Group. (21)

Type 5: The rest of lines (8, 9, ..., 20)

The lines of Type 1 and Type 2 are inputs and output of the circuit respectively, so they can take on any logical value over the range 0, 1, ..., p . Each line of Type 3 can have only two logical values, for example 0 and k if it is the output line from k – Group, and each line of Type 4 is fixed at one logical value. The rest of lines have the binary values (o, p).

Test generation

With the result of the above classification the equations that represent the complete test set will be derived on each case.

1) Type 1 lines

Theorem 1

The complete test set for a specific fault on the line of Type 1 is generated by analyzing the corresponding fault on the line of Type 5, i.e. the output line of $\overset{a\,b}{x}$ gate.

Proof; when one of the input lines of $\overset{a\,b}{x}$ gates is sfuck at a_i, this fault is indistinguishable from the s-a-p fault on the output line of that $\overset{a\,b}{x}$ gate if $a \leq a_i \leq b$, and it is indistinguishable from the s-a-o fault on that line if $a_i < a$ or $a_i > b$. Input tests for two indistinguishable faults are equal to each other. Hence, the proof is evident.

2) Type 2 line

Theorem 2

The complete test set for the s-a-a_i fault on line i of Type 2 is given by

$$\bar{f}_{a_i} + \sum_{a_i+1}^{p} f_j = p \qquad (12.a)$$

where $\bar{f}_{a_i\, a_i=0} = \bar{f}_0 = f_1 + f_2 + \cdots + f_p$

Proof; Since the Type 2 line is the output line ol the circuit, if this line is s-a-a_i, the output is always a_i for all input combinations. All input n tuples for which f=j where $j \neq a_i$ are the complete test set for this fault. $\bar{f}_{a_i}$=p is the input combinations that don't drive the output line to a_i. f_{a_i}=p is the input combinations that drive the output to a_i, but some of these may also drive the output to j where $j > a_i$ if DON'T CAREs are included. Therefore these input combinations must be added to the set of tests. The result is

$$\bar{f}_{aj} + f_{aj} \sum_{a_j+1}^{p} f_j = p \qquad (12.b)$$

It is well known from the Boolean algebra that for any x and y

$$\begin{aligned} \bar{x} + xy &= \bar{x}(1 + y) + xy \\ &= \bar{x} + \bar{x}y + xy \\ &= \bar{x} + y. \end{aligned} \qquad (13)$$

Hence, Eq.(12.b) can be simplified to Eq.(12.a).

The Type 2 line is the output line of the OR gate, thus the s-a-p fault on this line is indistinguishable from the s-a-p fault on the input lines of the OR gate (for example, 22 and 23 in Fig. 2).

Example 2; The complete test set for the s-a-1 fault on line 24 in Fig.2. is obtained from Theorem 2.

$$\overline{f}_1 + f_2 = p$$

$$({}^{0}x_1^{0}\,{}^{0}x_2^{0} + {}^{2}x_1^{2}\,{}^{1}x_2^{2}) + \overline{({}^{0}x_1^{0} + {}^{0}x_1^{1}\,{}^{0}x_2^{0})} = p$$

$${}^{0}x_1^{0}\,{}^{0}x_2^{0} + {}^{2}x_1^{2}\,{}^{1}x_2^{2} + \overline{{}^{0}x_1^{0}}\,\overline{{}^{0}x_1^{1}} + \overline{{}^{0}x_1^{0}}\,\overline{{}^{0}x_2^{0}} = p$$

The input combinations which satisfy this equation are $\{(0,0), (1,1), (1,2), (2,0), (2,1), (2,2)\}$. The corresponding decimal representation is $\{0,4,5,6,7,8\}$.

3) Type 3 lines

Theorem 3

The complete test set for the s-a-a_i fault on line i of Type 3 from k-Group is given by

$$f_k \prod_{k+1}^{p} \overline{f}_j = p, \qquad \text{if } a_i = 0 \qquad (14)$$

$$f_k \prod_{k+1}^{p} \overline{f}j + \pi_{a_i}^{p} \overline{f}_j = p, \qquad \text{if } 0 < a_i < k \qquad (15)$$

$$\prod_{a_i}^{p} \overline{f}_j = p, \qquad \text{if } a_i \geq k \qquad (16)$$

Proof; The line of Tye 3 from k-Group can have the logical values 0 and k when it is normal. Consider the following three cases.

Case 1 $a_i = 0$

We have to find out the input set for which $f_k = p$, that is line i should be made to have the logical value k. But some input combinations of this set may also drive the output line of the circuit to j where j>k if DON'T CAREs are included. Therefore these DON'T CAREs must be deleted from the test set. Hence the complete test set is given by Eq.(14). Case 2 $0<a_i<k$.

We have to find out the input set for which $f_k = 0$ or $f_k = p$. The input combinations for which $f_k = 0$ may drive the output line of the circuit to any logical value. But only the output values a_i, $a_i + 1, \ldots, p$ can appear at the output of the circuit because line i is stuck at a_i. Consequently, the input combinations for which $f = a_i, a_i+1, \ldots, p$ cannot detect the s-a-a_i fault on line i, and must be deleted from the test set. Next, the DON'T CAREs included in the input set for which $f_k = p$ drive the output line to j where j>k, and these also have to be deleted from the test set. Hence the complete test set is represented by

$$\overline{f}_k \prod_{a_i}^{p} \overline{f}_j + f_k \prod_{k+1}^{p} \overline{f}_j = p$$

Using Eq.(13) this equation is simplified to Eq.(15).

case 3 $a_i \geq k$

When $a_i = k$, we have to find out the input set for which $f_k = 0$. These input combinations may drive the output line to any logical value. Only the output values $k(= a_i)$, $k+1, \ldots, p$ can appear at the output because of the s-a-a_i fault on line i. Again, the input combinations for which $f = k, k+1, \ldots, p$ must be deleted from the test set. Thus,

$$\overline{f}_k \prod_{k+1}^{p} \overline{f}_j = \prod_{k}^{p} \overline{f}_j = \prod_{a_i}^{p} \overline{f}_j = p$$

when $a_i>k$, we have to find out the input set for which $f_k = 0$ or $f_k = p$. Under the condition $f_k = 0$, the input combinations for which $f = a_i, a_i+1, \ldots, p$ must be deleted from the test set. Under the condition $f_k = p$, the DON'T CAREs for which $f = a_i$, $a_i+1, \ldots, p$ must be deleted from the test set. Thus, the complete test set is given by

$$\overline{f}_k \cdot \prod_{a_i}^{p} \overline{f}_j + f_k \cdot \prod_{a_i}^{p} \overline{f}_j = p$$

$$(\overline{f}_k + f_k) \cdot \prod_{a_i}^{p} \overline{f}_j = \prod_{a_i}^{p} \overline{f}_j = p$$

Hence the theorem is proved.

Since the last gate of j-Group where j≠p is the partially enabled AND gate, the s-a-j fault on the output line of this AND gate is indistinguishable from the s-a-r fault on the input line of this AND gate where r≥j. And of course for the AND gate, any input s-a-0 and any output s-a-0 are the indistinguishable faults. The last gate of p-Group is the OR gate, thus input s-a-p and output s-a-p are indistinguishable faults.

Example 3; The complete test set for the s-a-0 fault on line 22 in Fig. 2. using Eq.(14) is

$$f_1 \cdot \overline{f}_2 = p$$

$$({}^{0}x^{0}_{1} + {}^{0}x^{1}_{2} \cdot {}^{0}x^{0}_{2}) \cdot (\overline{{}^{0}x^{0}_{1} \cdot {}^{0}x^{0}_{2} + {}^{2}x^{2}_{1} \cdot {}^{1}x^{2}_{2}}) = p$$

Thus we have

$$\{(0,0), (0,1), (0,2), (1,0)\} - \{(0,0), (2,1), (2,2)\}$$
$$= \{(0,1), (0,2), (1,0)\}$$

4) Type 4 lines

Theorem 4

The complete test set for the s-a-a_i fault on line i of Tyep 4 in k - Group is given by

$$f_k \cdot \prod_{k+1}^{p} \overline{f}_j = p, \text{ if } a_i < k \qquad (17)$$

$$f_k \cdot \prod_{a_i}^{p} \overline{f}_j = p, \text{ if } a_i > k \qquad (18)$$

Proof; this line is normally fixed at the logical vlaue k. There are two cases to consider. In both of the two cases, if the other input line of the AND gate is driven to 0 the output of the circuit becomes independent upon the fault on line i. Therefore, $f_k = p$ is the necessary condition to detect the fault on line i.

Case 1 $a_i < k$.

Under the condition $f_k = p$, the DON'T CARE's for which f = k+1, k+2, ..., p should be deleted from the test set. Hence the complete test set is given by Eq.(17).

Case 2 $a_i > k$

The DON'T CAREs for which $f = a_i, a_i+1, \ldots, p$ should be deleted from the test set. Hence the complete test set is given by Eq.(18).

Example 4; The complete test set for the s-a-2 fault on line 21 in Fig. 2. can be derived from Eq.(18).

$$f_1 \cdot \overline{f}_2 = p$$

This result is the same as in Example 3, But the output for the test set { (0,1), (0,2), (1,0) } is logic 2 in this example, while in Example 3 the output is logic 0.

5) Type 5 lines

Most of the lines in the circuit belong to Type 5. Since all the lines of Type 5 take on the binary values 0 and p, the Boolean difference can now be used to test faults on these lines.

Theorem 5

The complete test set for the s-a-a_i fault on line i of Type 5 in k-Group is given by

$$X_i \frac{dF_k}{dX_i} \prod_{k+1}^{p} \overline{f}_j = p, \qquad \text{if } a_i = 0 \qquad (19)$$

$$\frac{dF_k}{dX_i} \prod_{k+1}^{p} \overline{f}_j \left(\prod_{a_i}^{p} \overline{f}_j + X_i \right) = p, \text{ if } 0<a_i<k \qquad (20)$$

$$\overline{X}_i \frac{dF_k}{dX_i} \prod_{k+1}^{p} \overline{f}_j = p, \qquad \text{if } a_i \geq k \qquad (21)$$

Proof; There are three cases.

Case 1 $a_i = 0$

We must apply input patterns to drive the value of line i to p, while simultaneously making the output of k-Group dependent upon X_i. Some of these input combinations, DON'T CAREs, for which f = k+1, k+2, ..., p should be deleted from the test set. These conditions are met if and only if

$$X_i \frac{dF_k}{dX_i} \prod_{k+1}^{p} \overline{f}_j = p$$

Case 2 $0 < a_i < k$

The input combinations for which $\overline{X}_i \cdot dF_k/dX_i = p$ may drive the output line of the circuit to any logical value. But some input combinations for which $f = a_i, a_i+1, \ldots, p$ cannot detect the s-a-a_i fault on this line because these values appear at the output independent of the fault. The input set for which $X_i \cdot dF_k/dX_i = p$ may include DON'T CAREs, which have to be deleted from the test set. Thus we have

$$\overline{X_i}\ \frac{dF_k}{dX_i} \prod_{a_i}^{p} \overline{f}_j + X_i\ \frac{dF_k}{dX_i} \prod_{k+1}^{p} \overline{f}_j = p$$

$$\tag{22}$$

$$\frac{dF_k}{dX_i} \prod_{k+1}^{p} \overline{f}_j \left(\overline{X}_i \prod_{a_i}^{k} \overline{f}_j + X_i \right) = p$$

Using Eq.(13) this is simplified to Eq.(20).

Case 3 $a_i \geq k$

We have to find out the input set for which $\overline{X}_i \cdot dF_k/dX = p$. Again the input combinations for which $f = k+1, k+2, \ldots, p$ must be deleted from the test set. Thus Eq.(21) represents the complete test set for this fault.

When the logic fault a_i Where $a_i \geq k$ is made to propagate to the output of k-Group, it becomes k after passing through the partially enabled (k-enabled) AND gate. Therefore these faults are indistinguishable faults.

Example 5; The complete test set for the s-a-1 fault on line 11 in Fig.2 can be derived from Eq.(20).

$$\frac{dF_2}{dX_{11}} \left(\prod_{1}^{2} \overline{f}_j + X_{11} \right) = p$$

Subtituting

$$X_{11} = {}^{2}x_{1}^{2}$$

$$\frac{dF_2}{dX_{11}} = ({}^{0}x_{1}^{0}\ {}^{0}x_{2}^{0} + p \cdot {}^{1}x_{2}^{2}) \oplus ({}^{0}x_{1}^{0}\ {}^{0}x_{2}^{0} + 0 \cdot {}^{1}x_{2}^{2})$$

$$= {}^{1}x_{2}^{2}\ (\overline{{}^{0}x_{1}^{0}\ {}^{0}x_{2}^{0}}),$$

we have

$${}^{1}x_{2}^{2}\ (\overline{{}^{0}x_{1}^{0}\ {}^{0}x_{2}^{0}})\ (\overline{f}_1 \cdot \overline{f}_2 + {}^{2}x_{1}^{2}) = p$$

The result is $\{ (1,1), (1,2), (2,1), (2,2) \}$

If there is a NOT gate in the path from the faulty line to the output of k-Group, Theorem 5 should be modified because the value of this stuck-at fault is complemented by the NOT gate when it is sensitized to the output. In the implementation oriented algebra, the complement of a constant c is defined by $\overline{c} = p\text{-}c$. Therefore we have the following Lemma.

L

LEMMA

If there exist odd numbers of NOT gates along the path from the faulty line to the output, the complete test set for the s-a-a_i fault on this line i of Type 5 in k-Group is given by

$$\overline{X}_i\ \frac{dF_k}{dX_i} \prod_{k+1}^{p} \overline{f}_j = p, \text{ if } p - a_i = 0, \text{ i.e. } a_i = p \tag{23}$$

$$\overline{X}_i\ \frac{dF_k}{dX_i} \prod_{k+1}^{p} \overline{f}_j + X_i\ \frac{dF_k}{dX_i} \prod_{p\text{-}a_i}^{p} \overline{f}_j = p,$$

$$\text{if } 0 < p - a_i < k, \text{ i.e. } p - k < a_i < p \tag{24}$$

$$X_i\ \frac{dF_k}{dX_i} \prod_{k+1}^{p} \overline{f}_j = p, \text{ if } p - a_i \geq k, \text{ i.e. } 0 \leq a_i \leq p - k \tag{25}$$

Proof; The proof is similar to that of Theorem 5. Replacing X_i and a_i in Eqs.(19)(20)(21) by $\overline{X}_i$ and $p\text{-}a_i$ respectively, we have the above equations.

For the NOT gate, input s-a-a_i and output s-a-(p-a_i) are indistinguishable faults.

Fan Out

In Type 2, 3 and 4, there cannot exist fan-out lines. The test set for the fault on a fan-out line of Type 5 can be derived in the same way as the binary case if this line fans out to the lines in its own Group. The complete test set for this fault is thus represented by one of the equations in Theorem 5. However, if there is a NOT gate in the path to the output from one of the lines to which the fanout line fans out, the complete test set cannot be derived from Theorem 5. Consider Fig. 3. where line i fans out to

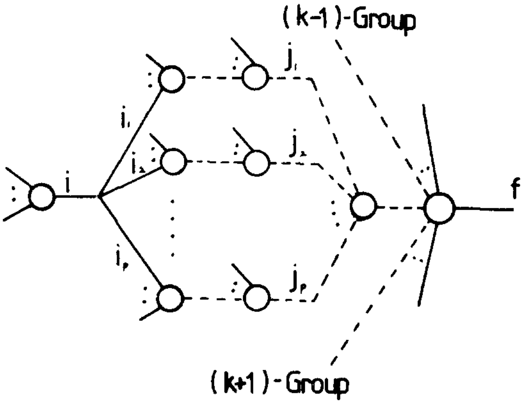

Fig. 3. Schematic of multivalued logic circuit with fan-out line i in k-Group.

lines $i_1, i_2, \ldots, i_p$ in k-Group. If there are odd numbers of the NOT gates in the path from line i_2 tc june j_2, the value of the fault on line i is complemented if it is sensitized along this path, but not complemented if it is sensitized along other paths. It is very difficult to predict what logical value will appear at the output of k-Group because of the fault on line i, and consequently difficult to know the effect of this fault on other Groups' output. In order to simplify fault detection, it is preferable to avoid the above situation when a circuit is designed. In fact, in this algebra any function or subfunction can be realized into a circuit without the NOT gate. For example, a subfunction ($\overline{{}^{0}x_1^{0}\,{}^{1}x_2^{1}}$) in ternary logic system is equal to ${}^{1}x_1^{2}\,({}^{0}x_2^{0} + {}^{2}x_2^{2})$.

When a line fans out to the lines in other Groups respectively, the complete test set for the fault on that line is obtained by summing the tests for the same fault on the lines to which that fanout line fans out even if there exist the NOT gates.

Theorem 6

Let i be a line of Type 5 which fans out to lines $i_1, \ldots, i_p$ in other Groups respectively, and let $T_i^{a_i}$ denotes the complete test set for the s-a-a_i fault on line i. Then,

$$T_i^{a_i} = T_{i_1}^{a_i} + T_{i_2}^{a_i} + \cdots\cdots + T_{i_p}^{a_i} \qquad (26)$$

Proof; If we apply an input combination of $T_i^{a_i}$ to the circuit with the s-a-a_i fault on line i, this fault is sensitized to the output along the path from line i via line i_j to the output. And it is obvious that an element of $T_i^{a_i}$ is also an element of a set among $T_{i_1}^{a_i}$, $T_{i_2}^{a_i}, \ldots, T_{i_p}^{a_i}$ Hence the theorem is proved.

Example 6; The complete test set for the fault of line 10 s-a-2 in Fig.2 can be obtained from Theorem 6. Here, only the result is shown.

$$\begin{aligned} T_{10}^{2} &= T_{15}^{2} + T_{16}^{2} \\ &= \{4,5\} + \{1,2\} \\ &= \{1,2,4,5\} \end{aligned}$$

There also exist fan-out lines in Type 1, for example line 1 and 2 in Fig.2. However, these lines are not direct input lines of ${}^{a}x^{b}$ gates, so they are assumed to have no faults in this paper.

Example 7; One of the minimal test set for the circuit of Fig. 2. is $\{0,1,3,4,6,7\}$, which can detect all possible logic faults in this circuit without testing it with every possible input and examining the output.

All the results in Examples 1, 2, ..., 7 are confirmed to be correct by comparing the truth tables of the normal and faulty circuits.

IV. Concluding Remark

In this paper we have used the Boolean difference concept to derive the equations of complete fault detection set. There exist network dependent indistinguishable faults which simplify fault detection but hamper fault location. However, the Allen-Givone implementation oriented algebra makes it easier to locate fault than the generalized ternary algebra because each Group is realized into a separate circuit module. The Boolean difference method gives us concise description of the many concepts involved in thest generation and these in turn provide valuable insight into the underlying processes and relationships. One may use these concepts to develop other methods of fault detection, such as D-algorithm. And it remains univestigated to explore whether these concepts can be generalized for use in multiple fault situation.

References

1. R.J. Spillman and S. Y. H. Su, "Detection of single, stuck-type failures in multivalued combinational networks," IEEE Trans. Comput., vol. C-26, pp. 1242-1250, December 1977.
2. Z. Vranesic, E. Lee, and K. C. Smith, "A many valued algebra for switching systems," IEEE Trans. Comput., vol. C-19, pp. 964-971, October 1970.
3. D. C. Rine, Computer Science and Multiple-Valued Logic. North-Holland, New York, 1977.
4. C. M. Allen and D. D. Givone, "A minimization technique for multiple-valued logic systems," IEEE Trans. Comput., vol. C-17, pp. 182-184, February 1968.

5.S. Y. H. Su and A. A. Sarris, "The relationship between multivalued switching algebra and Boolean algebra under different definitions of complement," IEEE Trans. Comput., vol. C-21, pp-479-485, May 1972.

6.M. Yoeli and G. Rosenfeld, "Logical design of ternary switching circuits," IEEE Trans. Electron. Comput., vol. EC-14, pp. 19-29, February 1965.

7.S. C. Lee, Modern Switching Theory and Digital Design. Prentice Hall, Englewood Cliffs, N. J., 1978.

8.A. Thayse and M. Davio, "Boolean differential calculus and its application to switching theory," IEEE Trans. Comput., vol. C-22, pp. 409-420, April 1973.

9.F. Sellers, M. Hasiao, and L. Bearnson, "Analyzing errors with the Boolean difference," IEEE Trans. Comput., vol. C-17, pp. 676-683, July 1968.

10.C. T. Ku and M. Masson, "The Boolean difference and multiple fault analysis," IEEE Trans. Comput., vol. C-24, pp. 62-71, January 1975.