Full Sensitivity and Test Generation for Multiple-Valued Logic Circuits

E. V. Dubrova, D. B. Gurov, J. C. Muzio

Department of Computer Science
University of Victoria
Victoria, B.C., Canada, V8W 3P6

Abstract

The notion of full sensitivity in a multiple-valued logic (MVL) circuit is introduced. A formalization of this notion using a specially defined operator, called mutual exclusion, is given. An expression of full sensitivity in the functional base of Rosser and Turquette is presented. The usefulness of this functional transformation with respect to test generation for MVL circuits is investigated.

1 Introduction

MVL circuits are hoped to solve several technological problems such as pin limitation and interconnection difficulties in VLSI circuit design. Several methods of ensuring reliability, and procedures for testing these circuits have been already presented. As in binary circuits, the single stuck-at fault model is the one most often investigated. The boolean differential calculus, which is used for the test generation of two-valued circuits, was appropriately generalized to MVL, although in different ways [2] [3] [5]. It was also observed, that the structure of value changes in the circuits under test has not been fully exploited, so many unnecessary calculations are made during test generation.

Usually, the task is to derive a minimal (or nearly minimal) complete test set. One starts with finding the complete test set for each fault of interest and applies a covering algorithm to these sets. Therefore, a test which is a member of the complete cover most probably detects many faults. And vice versa — any input vector detecting many faults has a good chance to become member of the complete cover. Intuitively, if there was some method for deriving input vectors which are tests for many faults, we could use the result of its application as a good starting point for some of the existing algorithms.

This paper is concerned with a special case in MVL circuits, which is particularly appropriate for deriving tests to detect many faults. This is the case, when the output of the circuit is made, by the application of some input vector, *fully sensitive* to some line l, i.e. each transition of this line from one logic value to another causes a change in the output logic value. For example, in a MVL circuit, consisting of one 2-input MAX gate only, the output of the gate is fully sensitive to one of its inputs when 0 is applied to the other input. Such an input vector will be a test for all single stuck-at-p faults on line l, where p is a logic value other than the fault-free one on line l under this input vector.

In Section 2 the notion of full sensitivity is formalized using a specially defined operator, called the mutual exclusion operator. The expression of full sensitivity in the functional base of Rosser and Turquette [1] and a procedure for its calculation are given in Section 3. The application of full sensitivity to test generation for MVL circuits is discussed in Section 4.

2 Mutual exclusion and full sensitivity

The work in this paper is based on a multiple-valued algebra, defined as follows.

Definition 2.1 *A multiple-valued algebra is called an algebra* $\mathcal{A} = \langle M; J, +, \cdot\,; 0, (m-1)\rangle$, *where*

(i) $M := \{0, 1, ..., (m-1)\}$ *is the totally ordered* carrier *of* $\mathcal{A}$;

(ii) $J := \{J_0, J_1, \ldots, J_{m-1}\}$ *is a set of* literal operators *such that*

$$J_i x := \begin{cases} m-1 & \text{if } x = i \\ 0 & \text{otherwise} \end{cases}$$

where x is a multiple-valued variable and $i \in M$ is a constant. For convenience, we write $J_i x$ as $\overset{i}{x}$;

(iii) " $+$ " *and* " $\cdot$ " *are the* binary operations *MAX and MIN, respectively;*

(iv) *0 and* $m-1$ *are* constants *of the algebra.*

A *complement* of a multiple-valued variable x is defined as $x' := (m-1) - x$, where " $-$ " is the regular arithmetic subtraction.

For the definition of full sensitivity we need a new operator, which is *decisive*, i.e. takes values in

0195-623X/94 $3.00

$\{0,(m-1)\}$ only, and has the ability of detecting whether its m arguments are pairwise distinct or not. We call this operator *mutual exclusion*. So:

Definition 2.2 Mutual exclusion *is the m-ary operator defined by*

$$mx(x_0,x_1,\ldots,x_{m-1}) := \begin{cases} m-1 & \text{if } x_i \neq x_j, i \neq j \\ 0 & \text{otherwise.} \end{cases}$$

where $i,j \in M$ *and* $x_0,x_1,\ldots,x_{m-1} \in M$.

If $m=2$, $mx(x_0,x_1) = x_0 \oplus x_1$, where $\oplus$ denotes sum modulo 2.

The mutual exclusion operator is extended over functions as usual. Obviously, this operator is independent of the order of its arguments. The literal operators can be expressed in terms of it as follows:

$$\overset{i}{x} = mx(0,\ldots,(i-1),x,(i+1),\ldots,(m-1)).$$

Using this operator we can define full sensitivity as a generalization of the Boolean difference.

Let $f(x_1,\ldots,x_n)$ denote a multiple-valued logic function of n variables of type $f: M^n \to M$. Further we use the following abbreviations: $\underline{x} := (x_1,\ldots,x_n)$ and $\underline{x}_i^k$ is the vector $\underline{x}$ with $x_i = k$, i.e.

$$\underline{x}_i^k := (x_1,\ldots,x_{i-1},k,x_{i+1},\ldots,x_n),$$

where $k \in M$, $i \in \{1,2,\ldots,n\}$.

Definition 2.3 *The* full sensitivity *of* $f(\underline{x})$ *with respect to* x_i *is defined as*

$$\frac{FSf(\underline{x})}{FSx_i} = mx[f(\underline{x}_i^0), f(\underline{x}_i^1), \ldots, f(\underline{x}_i^{m-1})]$$

where $i \in \{1,2,\ldots,n\}$.

From the definition, it follows that $\frac{FSf(\underline{x})}{FSx_i}$ does not depend on x_i. Further, it is decisive.

If $m=2$ then

$$\frac{FSf(\underline{x})}{FSx_i} = mx[f(\underline{x}_i^0), f(\underline{x}_i^1)] = f(\underline{x}_i^0) \oplus f(\underline{x}_i^1),$$

i.e. full sensitivity reduces to a boolean difference.

Example 1

Consider the following 3-valued function of two variables, given in a map form (Fig.1a). The full sensitivities $\frac{FSf(\underline{x})}{FSx_i}$, $i \in \{1,2\}$, may be found graphically in accordance with Definition 2.3 (Fig.1b,c). As shown on Fig.1b, $\frac{FSf(\underline{x})}{FSx_1}$ takes value 2 if the appropriate column of the map contains three different values, i.e. for $x_2 = 0$ and $x_2 = 1$. Similarly on Fig.1c $\frac{FSf(\underline{x})}{FSx_2}$ takes value 2 if the appropriate row of the map is a permutation of 0, 1 and 2, i.e. for $x_1 = 2$.

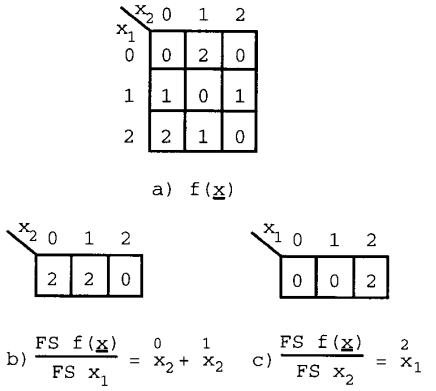

Figure 1: Maps for the function of the Example 1 and its full sensitivities.

3 Calculation of full sensitivities.

Definition 2.3 provides an easy way of finding the full sensitivity of small logic functions of 2 or 3 variables. But for larger functions we need some procedure, allowing us to calculate $\frac{FSf(\underline{x})}{FSx_i}$ algebraically. To describe such a procedure, we need to express mutual exclusion and full sensitivity in terms of the operators in $\mathcal{A}$. Let C be the set of all permutations over M, i.e.

$$C := \{(c_0,c_1,\ldots,c_{m-1}) \mid c_i \in M \wedge i \in M \wedge \\ \wedge\, mx(c_0,c_1,\ldots,c_{m-1}) = (m-1)\}.$$

For example, for $m=3$ $C := \{(0,1,2),(0,2,1),(1,0,2),(1,2,0),(2,0,1),(2,1,0)\}$.

The number of all permutations over M is m!, i.e. $|C| = m!$.

Property 3.1 *Let* $x_0,x_1,\ldots,x_{m-1}$ *be variables, such that for* $i \in M$, x_i *assumes any value from M.*

$$mx(x_0,x_1,\ldots,x_{m-1}) = \sum_{(c_0,\ldots,c_{m-1})\in C} \overset{c_0}{x_0} \cdot \ldots \cdot \overset{c_{m-1}}{x_{m-1}}$$

$$= \sum_{(c_0,\ldots,c_{m-1})\in C} \prod_{i\in M} \overset{c_i}{x_i}$$

Property 3.1 can easily be proved using Definition 2.2 and the definition of literals. □

For $m=3$

$$mx(x_0,x_1,x_2) = \overset{0}{x_0}\overset{1}{x_1}\overset{2}{x_2} + \overset{0}{x_0}\overset{2}{x_1}\overset{1}{x_2} + \overset{1}{x_0}\overset{0}{x_1}\overset{2}{x_2} + \overset{1}{x_0}\overset{2}{x_1}\overset{0}{x_2} + \overset{2}{x_0}\overset{0}{x_1}\overset{1}{x_2} + \overset{2}{x_0}\overset{1}{x_1}\overset{0}{x_2}.$$

Every MVL function of n-variables can be "partitioned" into m functions of $(n-1)$ variables using the

Table 1: Truth table for the function from Example 1.

x_1	x_2	$f(x_1,x_2)$	$\overset{0}{x}_1\, f(x_1^0)$	$\overset{1}{x}_1\, f(x_1^1)$	$\overset{2}{x}_1\, f(x_1^2)$	$\overset{0}{f}\,(x_1,x_2)$	$\overset{1}{f}\,(x_1,x_2)$	$\overset{2}{f}\,(x_1,x_2)$
0	0	0	0	0	0	2	0	0
0	1	2	2	0	0	0	0	2
0	2	0	0	0	0	2	0	0
1	0	1	0	1	0	0	2	0
1	1	0	0	0	0	2	0	0
1	2	1	0	1	0	0	2	0
2	0	2	0	0	2	0	0	2
2	1	1	0	0	1	0	2	0
2	2	0	0	0	0	2	0	0

generalized Shannon decomposition theorem (see [4]) as follows:

$$f(\underline{x}) = \overset{0}{x}_i\, f(\underline{x}_i^0) + \overset{1}{x}_i\, f(\underline{x}_i^1) + \ldots + \overset{m-1}{x}_i\, f(\underline{x}_i^{m-1}) \quad (1)$$

On the other hand we can write a MVL function in a Sum-of-Product form as consisting of m decisive functions $\overset{k}{f}\,(\underline{x})$, $k \in M$ (see [4]):

$$f(\underline{x}) = \sum_{k \in M} k \cdot \overset{k}{f}\,(\underline{x}) \quad (2)$$

where $\overset{k}{f}\,(\underline{x})$ is a shorthand for $(\overset{k}{f}(\underline{x}))$, i.e. a function which takes value $(m-1)$ for such $\underline{x}$ for which $f(\underline{x}) = k$, and 0 otherwise. Table 1 shows these two possible representations for the logic function from Example 1.

When we use Definition 2.3 to find full sensitivities we use the decomposition (1). But expression (2) provides us with more a convenient way for calculating $\frac{FSf(\underline{x})}{FSx_i}$. The following property shows that full sensitivities can be calculated as products of m functions.

Let $\overset{k}{f}\,(\underline{x}_i^p)$ be a shorthand for $(\overset{k}{f}(\underline{x}_i^p))$, i.e. a literal of the function $f(\underline{x}_i^p)$, $(k, p \in M)$. Further we use the following abbreviation:

$$\overset{k}{f}\,(\underline{x}_i^\phi) := \sum_{j \in M} \overset{k}{f}\,(\underline{x}_i^j).$$

Property 3.2

$$\frac{FSf(\underline{x})}{FSx_i} = \overset{0}{f}\,(\underline{x}_i^\phi) \cdot \overset{1}{f}\,(\underline{x}_i^\phi) \cdot \ldots \cdot \overset{m-1}{f}\,(\underline{x}_i^\phi) = \prod_{k \in M} \overset{k}{f}\,(\underline{x}_i^\phi).$$

Proof

$\frac{FSf(\underline{x})}{FSx_i} = mx(f(\underline{x}_i^0), f(\underline{x}_i^1), \ldots, f(\underline{x}_i^{m-1})$

{Df.2.3}

$= \sum_{(c_0,\ldots,c_{m-1}) \in C} \prod_{k \in M} \overset{c_k}{f}\,(\underline{x}_i^k)$

{Pr.3.1}

$= \sum_{(c_0,\ldots,c_{m-1}) \in C} \prod_{k \in M} \overset{k}{f}\,(\underline{x}_i^{c_k})$

{Commutativity of MIN}

$= \sum_{(c_0,\ldots,c_{m-1}) \in M^m} \prod_{k \in M} \overset{k}{f}\,(\underline{x}_i^{c_k})$

$\{a \neq b \rightarrow \overset{a}{f}\,(\underline{x}_i^j) \cdot \overset{b}{f}\,(\underline{x}_i^j) = 0\}$

$= \prod_{k \in M} \sum_{j \in M} \overset{k}{f}\,(\underline{x}_i^j)$

{Distrib. of MAX over MIN}

$= \prod_{k \in M} \overset{k}{f}\,(\underline{x}_i^\phi)$

{Def. of $\overset{k}{f}\,(\underline{x}_i^\phi)$}

□

There is a simple connection between the functions $\overset{k}{f}\,(\underline{x}_i^\phi)$ and $\overset{k}{f}\,(\underline{x})$. Expanding $\overset{k}{f}\,(\underline{x})$ using expression (1) we have:

$$\overset{k}{f}\,(\underline{x}) = \overset{0}{x}_i \overset{k}{f}\,(\underline{x}_i^0) + \overset{1}{x}_i \overset{k}{f}\,(\underline{x}_i^1) + \ldots + \overset{m-1}{x}_i \overset{k}{f}\,(\underline{x}_i^{m-1}).$$

Comparing this expression with the definition of $\overset{k}{f}(\underline{x}_i^{\phi})$ we can see that the functions $\overset{k}{f}(\underline{x}_i^{\phi})$ can be obtained from $\overset{k}{f}(\underline{x})$ by replacing symbolically all occurrences of the literals $\overset{j}{x}_i$ in the expression of $\overset{k}{f}(\underline{x})$ by $(m-1)$, $(i \in \{1,2,\ldots,n\},\ j,k \in M)$. This is the essence of the notation $\underline{x}_i^{\phi}$ — it means that all literals of the variable x_i "don't matter" and have to be removed.

So, the procedure of finding $\frac{FSf(\underline{x})}{FSx_i}$ for $i \in \{1,2,\ldots,n\}$ becomes the following:

1. Express the function $f(\underline{x})$ in its Sum-of-Product form (2), i.e.

$$f(\underline{x}) = \sum_{k \in M} k \cdot \overset{k}{f}(\underline{x});$$

Note that this expression must be in canonical Sum-of-Product form with no simplification using $i+j=j, (i<j)$

2. In each $\overset{k}{f}(\underline{x})$ replace symbolically all occurrences of the literal $\overset{j}{x}_i$ by $(m-1)$, $(j,k \in M)$;

3. Take the product of the functions obtained in step 2. The result of the multiplication is the full sensitivity $\frac{FSf(\underline{x})}{FSx_i}$.

Example 2

We compute $\frac{FSf(\underline{x})}{FSx_1}$ for the logic function from Example 1 using the procedure above.

1. The function has the following SOP form:
$f(\underline{x}) = 0(\overset{0}{x}_1\overset{0}{x}_2 + \overset{0}{x}_1\overset{2}{x}_2 + \overset{2}{x}_1\overset{2}{x}_2 + \overset{1}{x}_1\overset{1}{x}_2) + 1(\overset{1}{x}_1\overset{0}{x}_2 + \overset{1}{x}_1\overset{2}{x}_2 + \overset{2}{x}_1\overset{1}{x}_2) + 2(\overset{0}{x}_1\overset{1}{x}_2 + \overset{2}{x}_1\overset{0}{x}_2)$

2. The functions $\overset{0}{f}(\underline{x})$, $\overset{1}{f}(\underline{x})$ and $\overset{2}{f}(\underline{x})$ are:
$\overset{0}{f}(\underline{x}) = \overset{0}{x}_1\overset{0}{x}_2 + \overset{0}{x}_1\overset{2}{x}_2 + \overset{2}{x}_1\overset{2}{x}_2 + \overset{1}{x}_1\overset{1}{x}_2$
$\overset{1}{f}(\underline{x}) = \overset{1}{x}_1\overset{0}{x}_2 + \overset{1}{x}_1\overset{2}{x}_2 + \overset{2}{x}_1\overset{1}{x}_2$
$\overset{2}{f}(\underline{x}) = \overset{0}{x}_1\overset{1}{x}_2 + \overset{2}{x}_1\overset{0}{x}_2$

3. Replacing all occurrences of literals $\overset{0}{x}_1$, $\overset{1}{x}_1$ and $\overset{2}{x}_1$ by $(m-1)=2$ we have
$\overset{0}{f}(\underline{x}_1^{\phi}) = 2\overset{0}{x}_2 + 2\overset{2}{x}_2 + 2\overset{2}{x}_2 + 2\overset{1}{x}_2 = 2$
$\overset{1}{f}(\underline{x}_1^{\phi}) = 2\overset{0}{x}_2 + 2\overset{2}{x}_2 + 2\overset{1}{x}_2 = 2$
$\overset{2}{f}(\underline{x}_1^{\phi}) = 2\overset{1}{x}_2 + 2\overset{0}{x}_2 = \overset{1}{x}_2 + \overset{0}{x}_2$

4. Applying Property 3.2 we get
$\frac{FSf(\underline{x})}{FSx_1} = 2 \cdot 2 \cdot (\overset{0}{x}_2 + \overset{1}{x}_2) = \overset{0}{x}_2 + \overset{1}{x}_2$.

4 Full sensitivity in test generation.

Consider an internal line l of some multiple-valued combinational logic circuit, implementing the function $f(x_1,\ldots,x_n)$. To derive a test for a stuck-at fault on line l, line l is "cut" and is considered as a "pseudo-input" x_l (Fig.2).

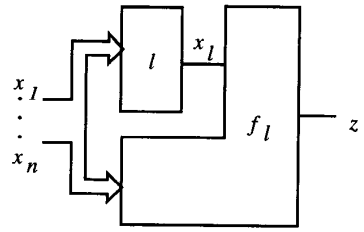

Figure 2: A multiple-valued combinational logic circuit.

The primary output z is expressed in terms of the primary input variables $x_1,\ldots,x_n$ and this pseudo-input x_l as $z = f_l(x_1,\ldots,x_n,x_l)$, where $x_l = l(x_1,\ldots,x_n)$. Let a_i be the value of the primary input variable x_i, for $i \in \{1,2,\ldots,n\}$. To find a test for the s-a-p fault on line l, $p \in M$, we need some set of values $(a_1,\ldots,a_n)$ such that $l(a_1,\ldots,a_n) \neq p$ and for some $k \neq p$ $f_l(a_1,\ldots,a_n,k) \neq f_l(a_1,\ldots,x_a,p)$. In our special case of full sensitivity we investigate the case when each transition on line l from one logic value to another logic value causes a change in the logic value of the primary output. So, we seek for values $(a_1,\ldots,a_n)$ and k, $a_i \in M, k \in M-\{p\}$ that satisfy the equation:

$$\overset{k}{l}(x_1,\ldots,x_n) \cdot \frac{FSf_l(x_1,\ldots,x_n,x_l)}{FSx_l} = (m-1) \quad (3)$$

Each solution $(a_1,\ldots,a_n)$ of the equation (3) is a test for a single s-a-p fault on line l, $p \neq k$. And since in a multiple-valued algebra there are $(m-1)$ values different from a given value $k \in M$, each $(a_1,\ldots,a_n)$ will be a test for $(m-1)$ single s-a-$\overline{k}$ faults on line l, where $\overline{k}$ denotes a value different from k, i.e. $\overline{k} \in M-\{k\}$. To find a complete test set for all m single stuck-at faults on a line we need to solve the equation (3) for two different values of k, for example for $k=k_1$ and $k=k_2$, $k_1,k_2 \in M$, $k_1 \neq k_2$. The first solution will give us the tests for $(m-1)$ single s-a-$\overline{k_1}$ faults, and the second — for a s-a-$\overline{k_2}$ faults, thus for the s-a-k_1 fault as well.

In general, to find a test set for all single stuck-at faults in the circuit there are as many full sensitivities to be computed, as there are lines in the circuit, and two equations of type (3) (for two different values of k) to be solved for each full sensitivity.

Example 3

Consider the 3-valued circuit given in Fig.3. We generate tests for single stuck-at faults on primary input 2 and on internal line 5, using full sensitivities.

The circuit is modeled by the 3-valued logic function

$$f(\underline{x}) := (x_1 + x_2x_3)(x_4 + x_2x_3)'.$$

For this function $\frac{FSf(\underline{x})}{FSx_2} = \overset{2}{x_1}\overset{2}{x_3}\overset{0}{x_4}$, and therefore for each $k \in \{0,1,2\}$ the vector $(2,k,2,0)$ is a test for all s-a-$\overline{k}$ faults on line 1. To find the tests for all single stuck-at faults on this line we only require to take two different values of k, for example 1 and 2, because the test $(2,1,2,0)$ will be a test for s-a-0 and s-a-2 faults on line 1, and the test $(2,2,2,0)$ will be a test for s-a-1 (and s-a-0) fault on line 1.

To find tests for single stuck-at faults on line 5 we express the function $f(\underline{x})$ in terms of the primary input variables $x_1,\ldots,x_4$ and the pseudo-input x_5:

$$f_5(x_1,\ldots,x_4,x_5) := (x_1 + x_5)(x_4 + x_5)'.$$

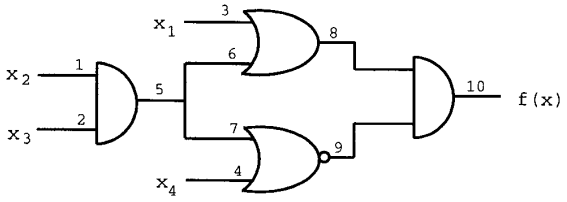

Figure 3: Circuit for the Example 3

For this function $\frac{FSf_5(x_1,\ldots,x_4,x_5)}{FSx_5} = \overset{2}{x_1}\overset{0}{x_4}$.

The actual value on line 5 depends on the values of x_2 and x_3. So, the equation (3) obtains the following form:

$$\overset{k}{(x_2x_3)}\overset{2}{x_1}\overset{0}{x_4} = 2.$$

Solving the equation for two different values of k, for example for $k = 1$ and $k = 2$ we obtain 4 tests, each detecting two stuck-at faults on line 5 (table 2).

Table 2: Set of tests for stuck-at faults on line 5.

k	test $x_1x_2x_3x_4$	stuck-at faults on line 5
1	2 1 1 0	s-a-0, s-a-2
	2 1 2 0	
	2 2 1 0	
2	2 2 2 0	s-a-0, s-a-1

A single s-a-p fault on line l is called *FS-detectable*, if equation (3) has at least one solution for some $k \in M-\{p\}$. A fault may not be FS-detectable, but be detectable in general, i.e. there exists a test detecting the fault. For example, tests for s-a-0 fault on line 6 in the circuit given on Fig.3 can not be obtained using full sensitivity, but we can find the tests for this fault using the methods presented in [2], [3] or [5]. Tests for FS-detectable faults, however, detect at least $(m-1)$ single stuck-at faults, which makes them good candidates for a minimal complete test set. Although the complexity of the method presented is high, there is a saving as against previous methods in that to find the tests for all single stuck-at faults on a line we only require two equations of type (3) to be solved, and not a separate equation for every fault on the line, as in [2], [3] or [5]. This observation suggests the following pro cedure: first, find tests for all FS-detectable faults as shown and, starting from this point, apply some of the conventional algorithms to derive tests for all FS-undetectable faults. Finally, apply a covering algorithm to obtain a minimal complete test set.

5 Conclusion

In this paper we have introduced the notion of full sensitivity of a MVL function and discussed its application to test generation for MVL circuits.

In further works full sensitivity could be expressed in terms of other complete functional bases like the one presented in [3]. It seems also to be worth investigating which classes of functions are advantageously manipulated by this approach.

Acknowledgments

The authors are indebted to Dr. Iwan G. Tabakow at HIMEE - Sofia, Bulgaria, for his suggestions concerning the search for appropriate generalizations of the Boolean difference.

References

[1] J.B.Rosser, A.R.Turquette, *Many-Valued Logic.*, 1952.

[2] M.Whitney, J.C.Muzio, *Decisive Differences and Partial Differences for Stuck-at Fault Detection in MVL Circuits*, 14th Int. Symp. MVL, 1984, pp.321-328.

[3] H.Lu, S.C.Lee, *Fault Detection in M-logic Circuits using the M-difference*, 14th Int. Symp. MVL, 1984, pp.62-70.

[4] J.C.Muzio, T.C.Wesselkamper, *Multiple-Valued Switching Theory*, Adam Hilger Ltd Bristol and Boston, 1986.

[5] T.A.Guima, M.A.Tapia, *Differential Calculus for Fault Detection in Multivalued Logic Networks*, 17th Int. Symp. MVL, 1987, pp.99-108.

[6] M.A.Tapia, T.A.Guima, A.Katbab, *Calculus for a Multivalued Logic Algebraic System*, Applied Mathematics & Computation, 1991, pp.225-285.