Multiple Fault Detection in a Network of Functions [1]

T. Raju Damarla, Mem. IEEE. [2] and M. Karpovsky, Sr. Mem. IEEE. [3]

ABSTRACT: In this paper we analyze the problem of multiple fault detection in a network of functions (cells implementing functions). It is shown that all multiple stuck-at-faults (s-a-fs) and all single bridging faults are detected by a test set which detects all multiple faults in every cell. In general it is shown that a number of test patterns is proportional to the number of lines in a network. For homogeneous trees the number of test patterns is shown to be linear with number of input variables. Upper bounds on number of test patterns is presented.

1 Introduction :

In this article we will investigate detection of multiple faults in a network of functions. It has become a common trend for implementing a system of complex Boolean function(s) in a single chip and then interconnecting various chips implementing different functions in order to realize a larger function or functions (see fig 1.). In the event, each cell representing a single gate, the network may be veiwed as a conventional implementation of a function f.

[1]This work was supported by the National Science Foundation under the Grant DCR-8317763.

[2]Dept. of Electrical Engineering, Univ. of Kentucky, Lexington, KY 40506.

[3]ECS Dept., Boston University, 110 Cummington st., Boston, MA 02215.

Several authors have investigated the problem of fault detection [1-7] in combinational logic networks. Some have investigated multiple fault detection in combinational networks [2-4]. In analyzing the fault detection probelm, different topologies have been studied by various authors for networks implementing a single Boolean function. Hayes [1] has considered fault detection in Uniform NAND trees. Gault and et all [2] have investigated multple fault detection in special classes of trees with input fanout. It was shown [2] that if a tree consists of AND and OR gates in alternate levels, then all multiple stuck-at-faults (s-a-fs) are detected by a test set which detects all single s-a-fs. It is also shown that any Boolean function can be realized as a tree with input fanout. Kohavi, et all [6] have investigated fault diagnosis in combinational tree networks.

In this paper we will consider a much general fault model which is given below. We will show that an upper bound on a number of test patterns which detect all multiple s-a-fs is proportinal to the number of cells. If each cell implements a single cell, then a number of test patterns is shown to be proportional to the number of lines in the network.

It will be assumed that the number of input lines to a cell c_i is m_i. We assume that any function f_i is implemented by combinational logic only, and each cell implements only one function. The fault model considered for testing a network is given below.

Fault Model: A fault may occur in any single cell and fault can be a single or multiple stuck-at-fault (s-a-f) and single bridging fault. It is assumed that a fault does not result in sequential behaviour. Multiple s-a-fs may occur in the inter connecting lines.

Since the fault model is very general, in order to detect the faults in any cell, one has to apply all 2^{m_i} different input vectors to a cell c_i, $i \in \{1, 2, ...\}$

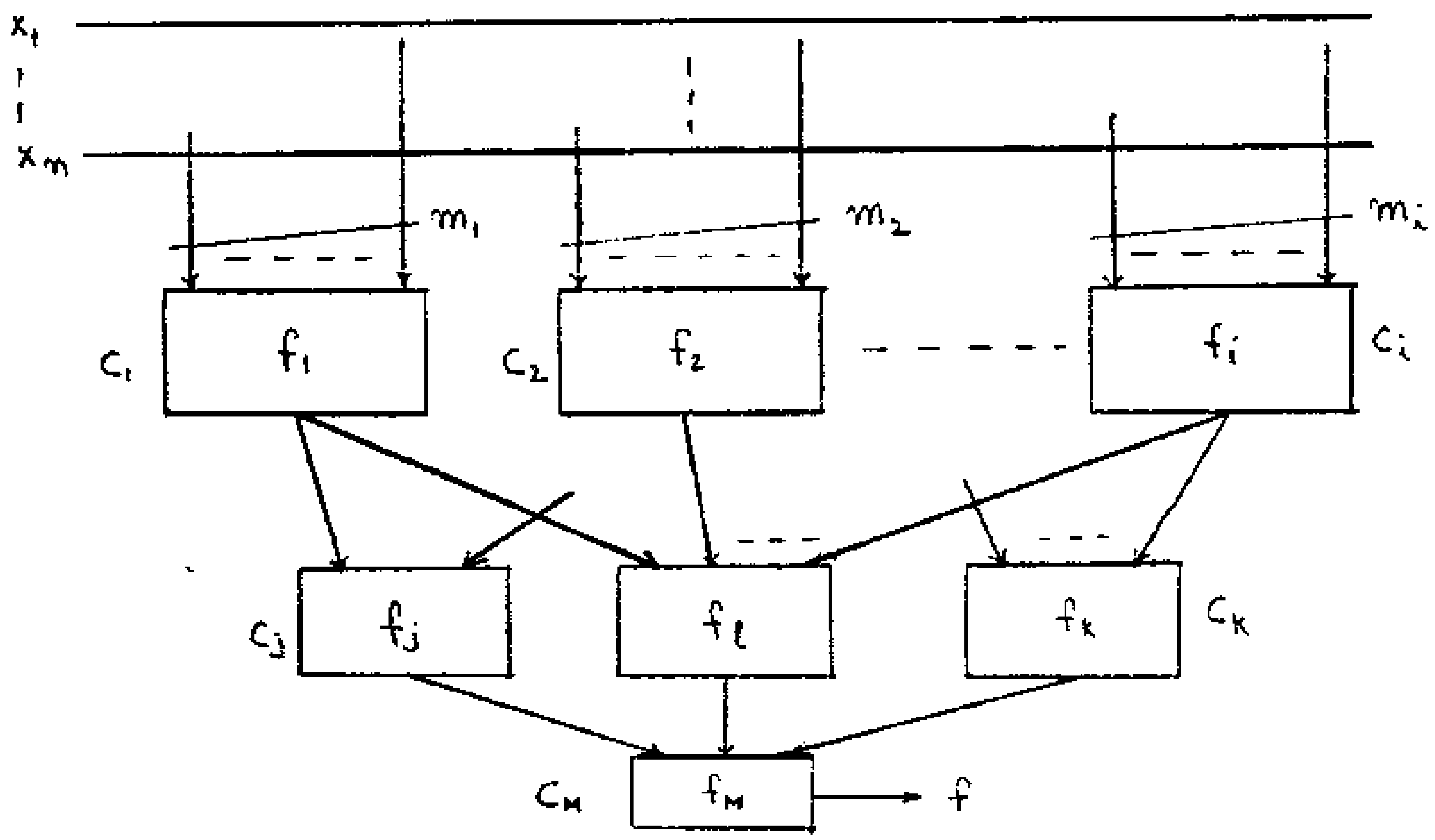

Figure 1: Network of Functions.

(exhaustive testing of each cell) and at the same time the conditions should be such that a fault in any cell propagates to the primary output line, which is the only observation point in the whole network.

2 Network of Functions :

In this section, we will consider a network of functions. General structure of a netowrk of functions is shown in fig 1. Through out this paper it will be assumed that a network is irredundant and hence all single s-a-fs can be detected.

Now, if h_i is a line in a network implementing a function $f(X) = f(x_1, x_2, ..., x_n)$, then a stuck-at-0 (s-a-0) fault at h_i (denoted as $h_i/0$) can be detected [7] by any vector X such that

$$h_i \frac{df}{dh_i} = 1. \tag{1}$$

Similarly, a stuck-at-1 (s-a-1) fault at h_i (denoted as $h_i/1$) can be detected by another vector X', where

$$\bar{h}_i \frac{df}{dh_i} = 1. \tag{2}$$

If $\frac{df}{dh_i} = 1$, then the line h_i is said to be sensitized, in other words, a s-a-f at h_i would propagate to the output f.

2.1 Generation of Test Patterns :

It is well known [8] that test pattern generation for general combinational networks is a NP-complete problem. We will not give actual algorithm for generating test patterns, but we will formulate the conditions for fault propagation that must be met by any test pattern if it has to test certain faults that may occur in a network.

Let $f(x_1, x_2, ..., x_n)$ be a function that is implemented by a network (fig 1.), and let $f_i = f_i(x_{i1}, x_{i2}, ..., x_{im_i})$ be a function implemented by a cell c_i in the network. Let us consider a subset of test patterns $\{T_i\}$ which detect all faults in a cell c_i, where

$$T_i = \{t_{i1}, t_{i2}, ...\},$$

and

$$t_{ij} = (\dot{x}_1, \dot{x}_2, ..., \dot{x}_n),$$

where $\dot{x}_k \in \{0, 1\}$ and $|T_i|$ indicates the number of test patterns in $\{T_i\}$. Then T_i must satisfy the following conditions :

1. If a test pattern t_{ik} detects a single s-a-f at input line x_{ij}, then

$$\frac{df_i}{dx_{ij}} = 1, \quad and$$

2.

$$\frac{df}{df_i} = 1.$$

The first condition ensures that a s-a-f at x_{ij} would propagate to the cell's output f_i and the second condition propagates the fault to the final output point f. It is clear that if a s-a-f at x_{ij} is detectable then the above conditions must be satisfied by a test pattern. Such a test pattern must exist in 2^n possible input vectors which would detect a fault if the fault is detectable in the cell c_i. Now consider the following heuristic algorithm :

ALGORITHM-1:

- Step 1: Consider a cell c_i and generate an exhaustive test set T_i which satisfy the conditions mentioned above regarding the propagation of faults. Note that, due to fanout nature of a network it may not be possible to generate all 2^{m_i} number of test patterns for a cell c_i .

- Step 2: If c_j is in the path of c_i and if $t_{ik} \in \{T_i\}$ sensitizes c_j, then consider t_{ik} as one of the elements of test set T_j for the cell c_j. While generating test set for a cell c_i, assignment of the input variables should be such that as many cells other than c_i as possible should be desensitized (optimal desensitization [1]).

- Step 3: Repeat Step 1 and 2 for all i.

At this juncture, it is worth while discussing the fault detection capabilities of the test set constructed according to the above algorithm. The faults in a network may be catagorized as :

- i. Input and Internal Faults in a cell c_i, for all i.

- ii. Faults at Inter Connecting Lines.

i. Input and Internal Faults in a cell c_i.

Since, every cell c_i is tested exhaustively by applying all the possible 2^{m_i} test patterns, all the input and internal faults are detected. If only fewer test patterns than 2^{m_i} are generated, due to fanout nature of a network, and if this test set T_i is not sufficient to detect all possible faults, then those faults are undetectable faults, since only $|T_i|$ test patterns out of 2^n can be applied to c_i. In otherwords, even the test set which tests the whole network exhaustively by applying all 2^n test patterns will not detect those faults.

ii. Faults at Inter Connecting Lines.

All single s-a-fs at input line of a cell c_i, $i = (1, 2, ..., M)$, are detected by a test set $\{T_i\}$, since the network is irredundant.

Now, consider multiple s-a-fs at lines x_{im} and x_{kl}. While testing cell c_i, the following cases may happen.

Case 1: If test patterns which detect s-a-fs at x_{im} sensitizes only x_{im} (that is $\frac{df}{dx_{im}} = 1$) but not x_{kl} then only single fault is effective and the fault will be detected by one of the test patterns in $\{T_i\}$.

Case 2: If both lines x_{im} and x_{kl} are sensitized by a test pattern then the multiple fault may not be detected by a test pattern which sensitizes both the lines, since the multiple faults may mask each other. However, we will show that there exists a test pattern in a test set generated by Algorithm-1 which would detect the multiple fault.

Let x_{im} be an input to a cell c_i and x_{kl} input to c_k. If c_i and c_k are entirely independent, then Case-1 is applicable and all multiple faults are detectable. If c_i and c_k are in the fanout paths of some cell then one of the two possibilities may happen.

Case 2i: x_{im} and x_{kl} may be different fanout lines of a cell's output (see fig 2.1a.). If this is the case, and due to irredundant nature of the network, when c_i is tested

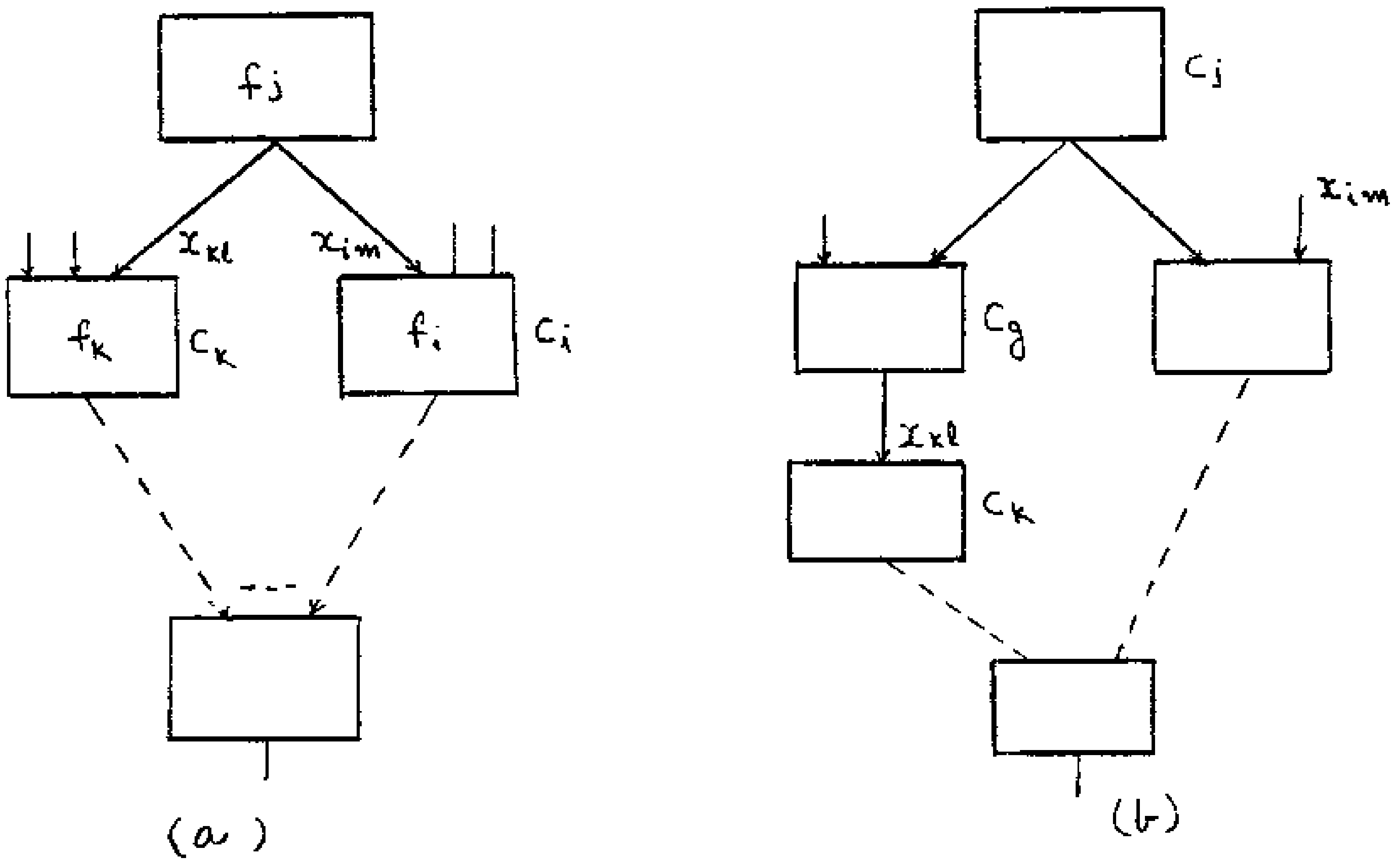

Figure 2: Multiple s-a-fs at x_{im} and x_{kl}.

exhaustively, there exists test patterns which sensitize either c_i or c_k but not both. Then, depending on which one of the cells sensitized only x_{im} or x_{kl} is sensitized and fault is detected, since all s-a-fs are detectable in an irredundant network. Note that generalization for more than two fanouts from c_j is trivial.

Case 2ii: Let x_{im} and x_{kl} are not the fanout lines of an output of a cell c_j as shown in fig 2.1b. If a multiple fault at x_{im} and x_{kl} is detectable then there exists a test which sensitizes c_g but not c_i. It is also evident that there exists at least one input x_{gp} to the cell c_g which is not input to c_i. Hence, the test patterns which detect $x_{gp}/0$ or $x_{gp}/1$ would detect the s-a-fs at x_{kl}, which is the output of the cell c_g.

If the fault is at the input of c_k but not at the output of c_g, still the fault is similarly detected during exhaustive testing of c_k while c_i is not sensitized.

The above results are presented in the following theorem.

Theorem 1 *: In a network constructed with cell implementing functions* $f_i(x_{i1}, x_{i2}, ..., x_{im_i})$, $i = (1, 2, ..., M)$, *all the single and multiple faults in any cell* *and all multiple s-a-fs at the inter connecting lines are detected by a test set T, and*

$$|T| \leq \sum_{i=1}^{M} 2^{m_i} - 2, \tag{3}$$

number of test patterns, where M is the number of cells in a network.

Proof: Detection of faults is discussed above. In order to test a cell c_i exhaustively one would require 2^{m_i} test patterns. However, while testing c_i, some of the cells which are in the path between c_i and the final output are also tested. So for these intermediate cells some test patterns need not be repeated. In the worst case, if there are no cells in between except the final output cell; for this output cell at least two of the test patterns which produce output 0 and 1 need not be applied. And hence the number of test patterns is upper bounded by $\sum_{i=1}^{M} 2^{m_i} - 2$.

Q.E.D.

It is interesting to note that all bridging faults among the inter connecting lines are also detected. This is true since any test set which detects all multiple s-a-fs should indeed detect the bridging faults. The reason being that a bridging fault may be veiwed as a special case of s-a-fs. For example, if lines x_{im} and x_{kl} are bridged, then that fault can be detected by a test pattern which detects multiple s-a-fs $x_{im}/0$ and $x_{kl}/1$ (or $x_{im}/1$ and $x_{kl}/0$).

From the above discussion, we may conclude that for almost all practical purposes, almost all faults in a network can be detected by a test set which exhaustively tests individual cells in a network. One can acheive the same results by testing the whole network exhaustively by applying all 2^n test patterns, where n is the number of input variables to a network. The following example would illustrate the difference between $|T|$ and 2^n.

Example 1 *: Let $n = 20, M = 1000$ and let us assume that every cell has 8 input lines. Then*

$$Exhaustive\ testing\ requires\ 2^n = 1048580 \quad test\ patterns$$

$$|T| = 1000 \,.\, 2^8 - 2 = 255998.$$

The above example and (3) have illustrated the dependency of $|T|$ on M and the number of inputs to a cell. While the dependency on the former is linear, the number of test patterns grow exponentially with number of inputs to a cell. In Section 4, we will present techniques to reduce the number of test patterns by redesigning each cell as a combination of cells with fewer inputs.

2.2 Networks with Multiple Outputs:

In this section we will consider networks with multiple outputs. Once again we assume every cell in a network is a single output cell. The general structure of a multiple output network is shown in fig 3. It is worth noting that programmable logic array structures are a special case of networks considered here with only two levels.

Generation of test patterns is done for the networks with multiple outputs as per Algorithm-1 by considering one output at a time. Clearly, if some cells are covered by a function O_i, then they need not be considered while generating test patterns with another function $O_{k \neq i}$. Test generation procedure ends when exhaustive test sets for all the cells in a network are generated. Once again the number of test patterns which would detect all multiple s-a-fs and single bridging faults is upper bounded by

$$|T| \leq \sum_{i=1}^{M} 2^{m_i} - 2.$$

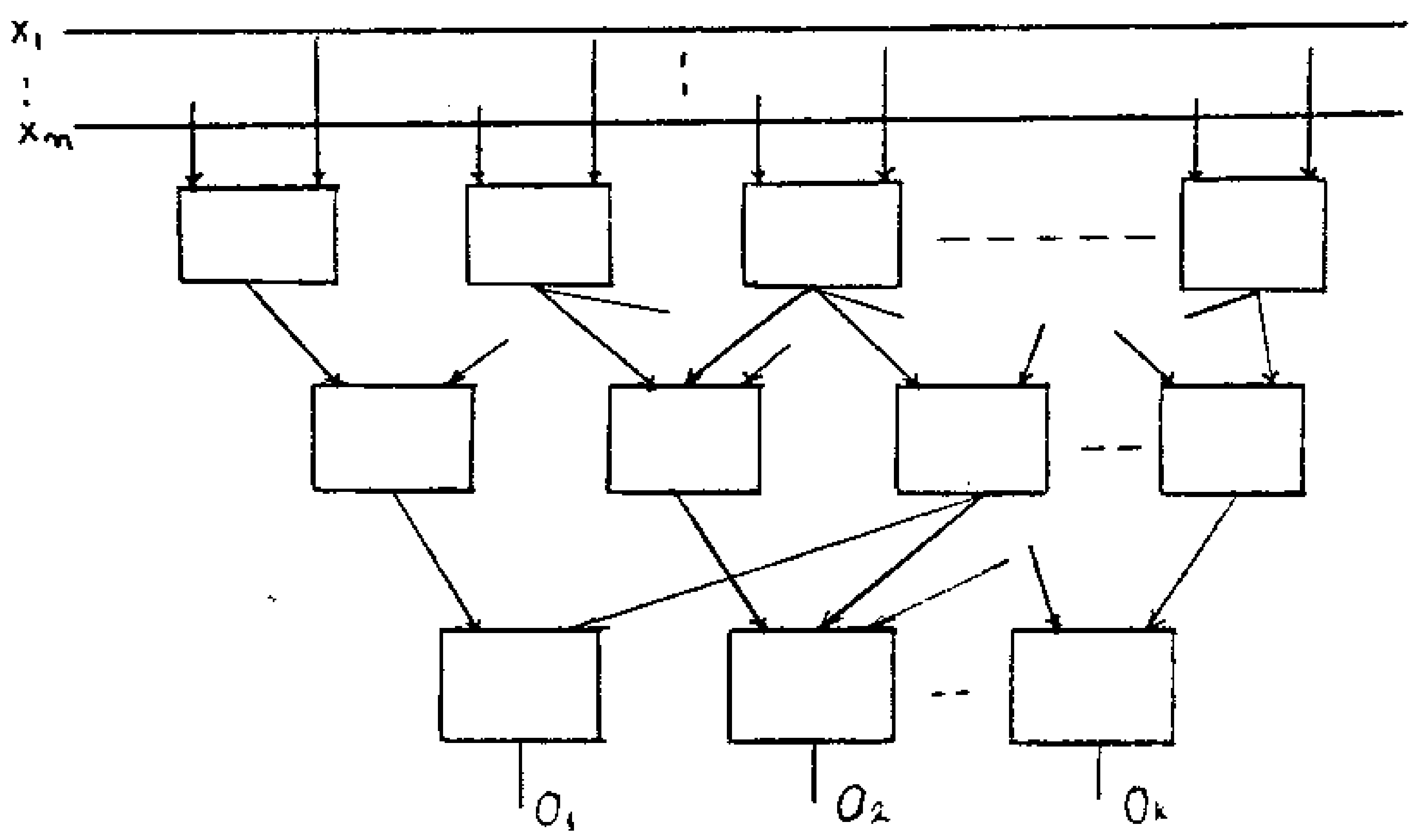

Figure 3: General Structure of a Multiple Output Network.

Multiple s-a-f and single bridging fault detection in multiple output networks is similar to the case of detection of multiple s-a-fs in a single output network. Single bridging faults among the output lines is done by a test pattern which assigns $O_i = 1$ and $O_k = 0$ and such a test pattern always exists if $O_i(X) \neq O_k(X)$, $\forall X$, which is the case for an irredundant network.

3 General Trees :

In this section, general trees constructed using cells implementing functions f_i, $i \in \{1, 2, ...\}$ will be considered. The general structure of a tree is shown in fig 4.

Suppose if the network consists of only two levels then any fault in a cell c_i in the second level would manifests at its output which is connected to the input of the output cell (see fig 5). Without loss of generality let this input be z_i. A

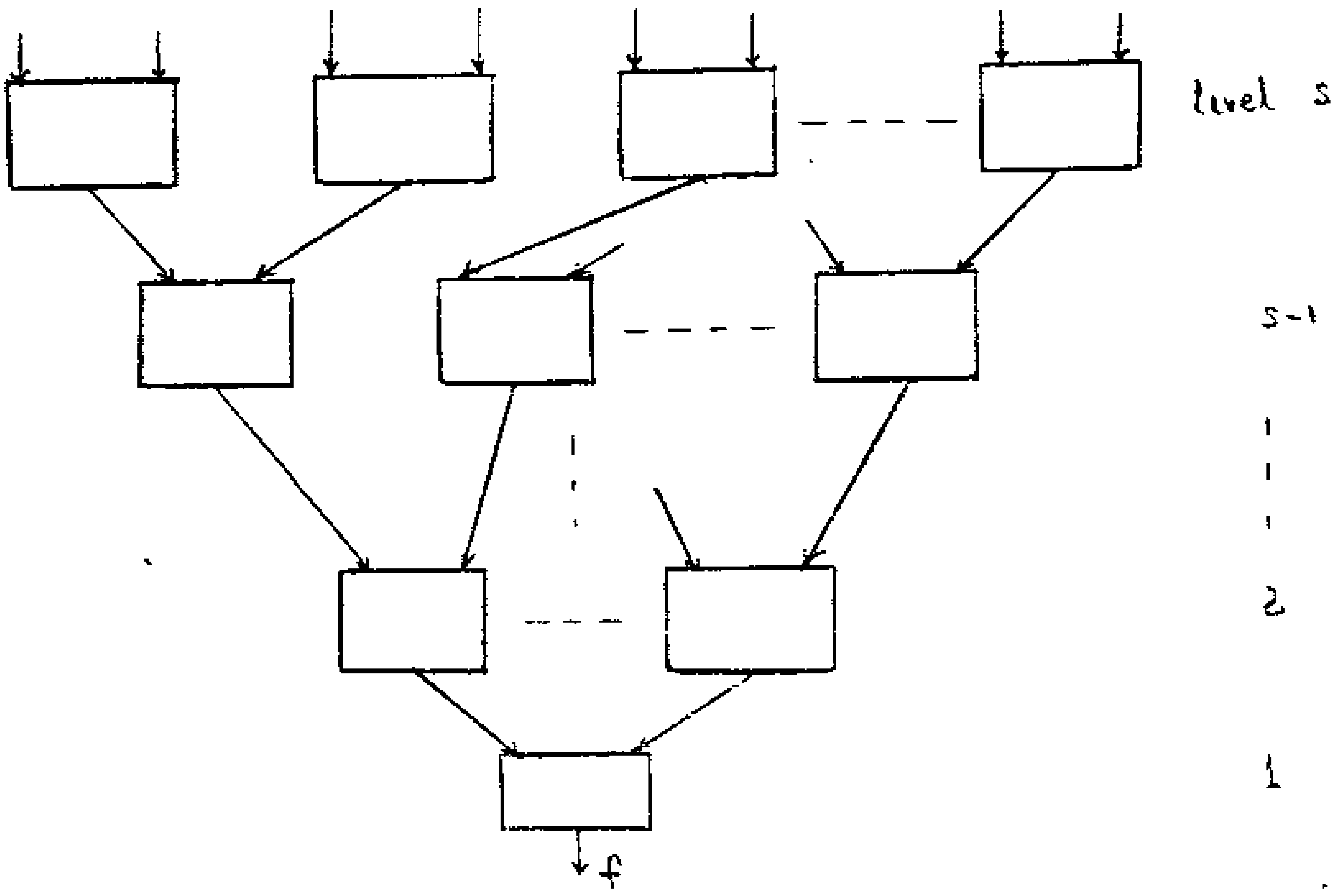

Figure 4: General Structure of a Tree of Functions.

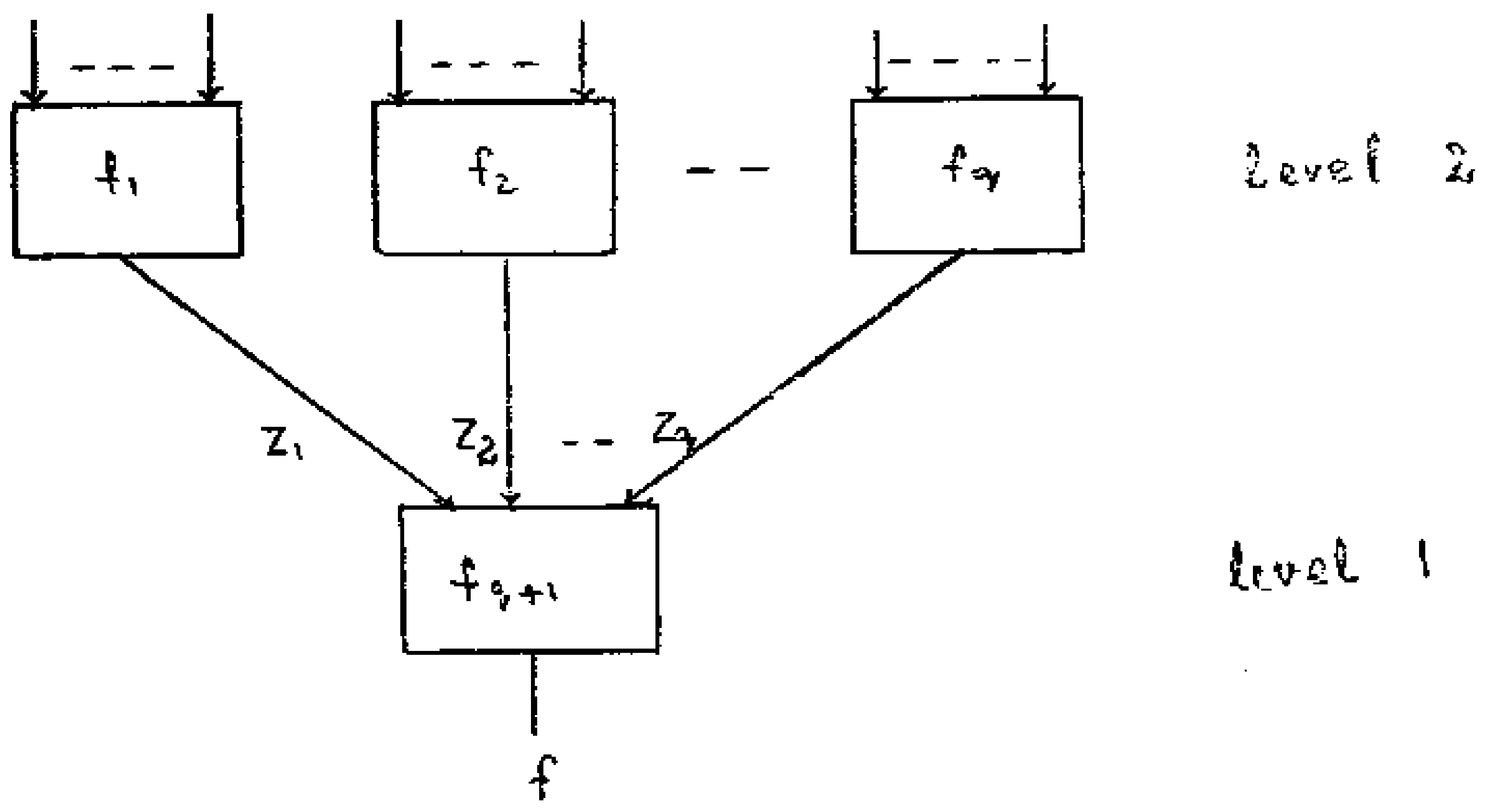

Figure 5: A Two-level Tree.

change in the input z_i would propagate to the output F if [Sellers, et all]

$$\frac{df}{dz_i} = 1. \tag{4}$$

If a vector Z is such that $\bar{z}_i \frac{df}{dz_i} = 1$, then $z_i/1$ (z_i stuck-at-1) fault can be detected, similarly if Z satisfies $z_i \frac{df}{dz_i} = 1$, then $z_i/0$ (z_i stuck-at-0) fault can be detected. However if a vector Z is such that

$$z_i\frac{df}{dz_i} = z_j\frac{df}{dz_i} = 1 \tag{5}$$

then both $z_i/0$ and $z_j/0$ can be detected by Z. While maintaining the condition (5) one can apply the input set of vectors to the cells c_i and c_j (with output lines z_i and z_j) which results in a '1' at their outputs (ie., $z_i = 1$ and $z_j = 1$). So if there is a fault in either c_i or c_j, then a fault manifests itself as either $z_i/0$ or $z_j/0$ and hence the output F would change detecting the fault.

The testing methodology for a multilevel tree, presented subsequently will be based on testing one cell at a time exhaustively rather than testing multiple cells at a time. The following definition would be useful.

Definition: Let

$$F_j^i = \{X | f_j(X) = i\}, \ \ i \in \{0,1\}.$$

Clearly $|F_j^0| + |F_j^1| = 2^{m_j}$, where m_j is the number of input variables to a cell c_j.

Test generation procedure will now be described for a two level tree.

3.1 Test Pattern Generation For A Two Level Tree:

First test patterns which test the cells in the second level will be generated and then the test patterns required to test the output cell will be generated.

Algorithm-2:

- Step 1: Select a vector Z such that $z_1 \frac{df_{q+1}}{dx_1} = 1$, (see fig 5), apply one of the vectors in F_1^1 to cell c_1 in the second level and any vector from to $F_j^1(F_j^0)$, $j \neq 1$, if $z_j = 1(0)$.

- Step 2: Repeat step 1, till all the vectors from F_1^1 are applied to cell c_1 in second level, for the same Z.

- Step 3: Repeat steps 1 and 2 with Z such that $\bar{z}_1 \frac{df_{q+1}}{dx_1} = 1$ (note that this Z differs from the previous one only in z_1), and apply one of the vectors in F_1^0 to cell c_1.

- Step 4: Step 1 to 3 are repeated for all z_i, $i \in \{1, ..., q\}$

- Step 5: Only cell for which exhaustive testing is not done is the cell c_{q+1}. However some of the test patterns are already applied while testing the cells in the second level. Apply the remaining test patterns to cell c_{q+1}.

Let the total number of test patterns thus generated is given by N_2

$$N_2 = \sum_{i=1}^{q} |F_i^1| + \sum_{i=1}^{q} |F_i^0| + |F_{q+1}^1| + |F_{q+1}^0| - 2 = \sum_{i=1}^{q+1} 2^{m_i} - 2. \tag{6}$$

where the subscript in N_2 indicates the number of levels, and -2 is due to the fact that at least one of the vectors from $|F_{q+1}^1|$ and $|F_{q+1}^0|$ should have been applied while testing the cells in second level.

The above test pattern generation process can be extended for test pattern genertion for a s-level network, in a straight forward fashion as shown in fig 6.

Definition: A tree is called homogeneous if all the cells have same number of input lines.

Definition: A cell is called a primary cell if the inputs to this cell are all primary input variables, all other cells are called secondary cells.

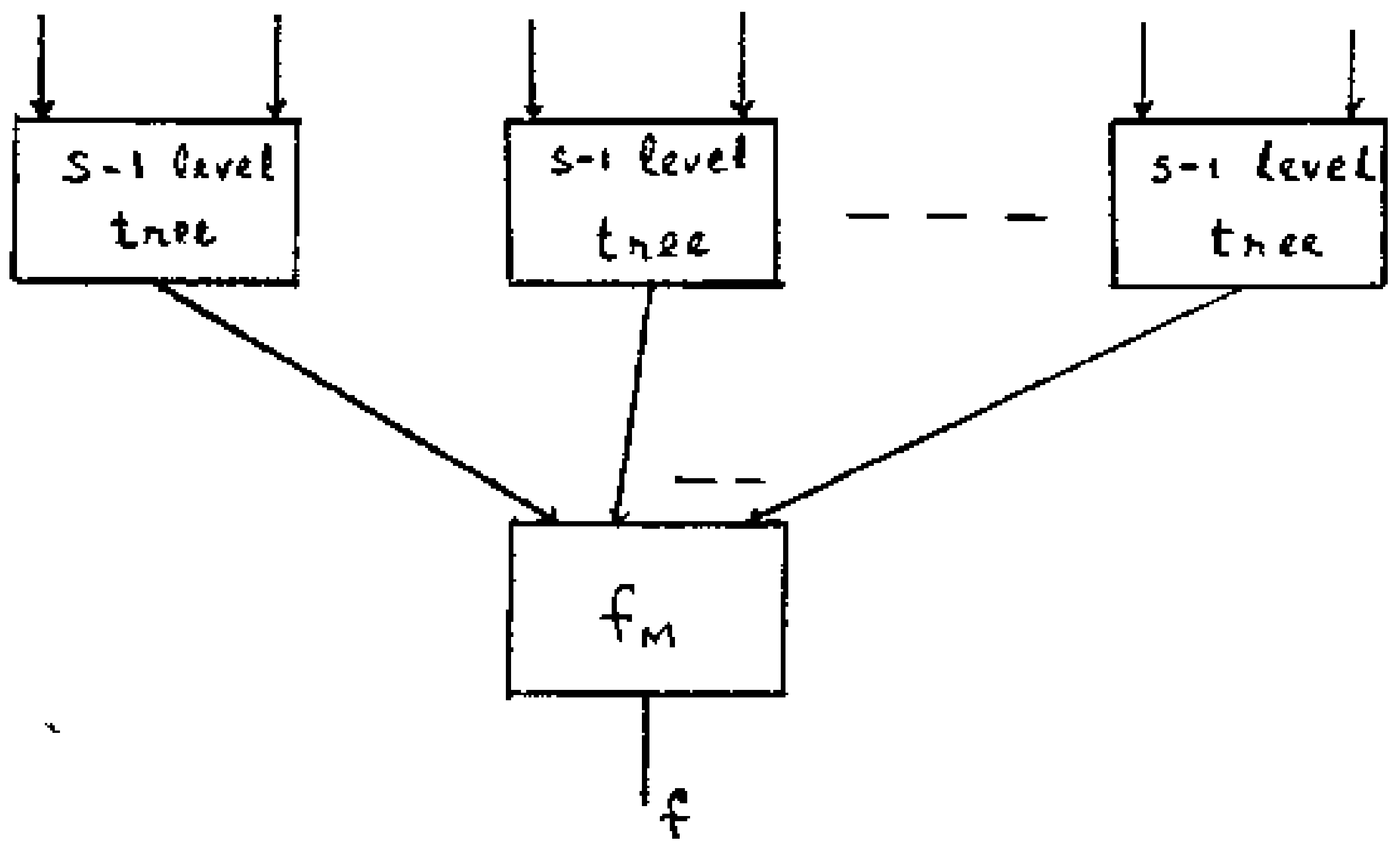

Figure 6: An s-level Tree represented as a Two-level Tree.

In a homogeneous tree, if every cell has m-inputs, and if the number of levels in the tree is s, then a recursive formula for N_s is given by

$$N_s = mN_{s-1} + 2^m - 2, \quad s > 1, \; N_1 = 2^m. \tag{7}$$

Solving equation (7) one would obtain

$$N_s = 2^m \frac{m^s - 1}{m - 1} - 2\,\frac{m^{s-1} - 1}{m - 1}. \tag{8}$$

It is interesting to know that the first number in (8) represents total number of cells in a tree of s-levels and the second part corresponds to the number of secondary cells. This equation (8) is exactly same as equation (3) but for the second part.

Lemma 1 : *Test patterns generated by the above algorithm would detect all multiple s-a-fs at the inter connecting lines and all faults inside a cell.*

Proof: Multiple s-a-fs at the input of any single cell are detected since each cell is tested exhaustively.

Case-i: Let x_i and x_j are faulty, where x_i is an input to a cell c_i and x_j is input to c_j.

i) If Z (which is used for generating test paatterns for c_i) is such that $z_i \frac{df_{q+1}}{dz_i} = z_j \frac{df_{q+1}}{dz_j} = 1$ and the fault distorts either z_i or z_j but not both, then fault is detected, since fault propagates to the output.

ii) If the fault distorts both z_i and z_j, then test patterns generated by Z' (Z' and Z differ in only in z_i) for c_i where $\bar{z}_i \frac{df_{q+1}}{dz_i} = z_j \frac{df_{q+1}}{dz_j} = 1$ results in distorting of only z_j and hence fault is detected.

Case-ii: If z_i and x_j are fault, then this fault can be detected while testing c_{q+1} exhaustively, since the fault amounts to single or multiple faults at the input of the cell c_{q+1}.

Case-iii: Faults inside a cell are always detected because each cell is tested exhaustively.

Q.E.D.

The above lemma holds for any tree with arbitrary number of levels since any multi level tree can be represented as a two level tree as shown in fig 6.

At this point it is in order to mention that the number of test patterns generated by Algorithm-2 is not minimal. One can generate optimal number of test patterns by choosing a vector Z with maximum number of variables $z_{i1}, z_{i2}, ..., z_{il}$, such that

$$z_{i1} \frac{df_{q+1}}{dz_{i1}} = ... = z_{il} \frac{df_{q+1}}{dz_{il}} = 1,$$

and generating test patterns which would simultaneously test cells $c_{i1}, ..., c_{il}$, unlike testing one cell at a time as in the case of Algorithm-2. However, if we adopt optimal test pattern generation, it will not guarantee detection of multiple s-a-fs at inter connecting lines.

Even though, Algorithm-2 does not generate optimal test set, the number of test patterns is

$$N_s < C.n, \tag{9}$$

where C is a constant and n is the number of primary input lines, if s is large and m_i, $\forall i$, is small (as can be evidenced by (8) when $m_i = m = const$, $s \longrightarrow \infty$).

4 Minimization of Number of Test Patterns:

Consider equations (3) and (8). In both the cases, the number of test patterns grow exponentially with a number of input lines to a cell. Consider a case where a function is implemented by a single cell with nine input lines as shown in fig 4a. In order to test this network exhaustively one would require 512 test patterns. However, if the same function can be implemented as two level network as shown in fig 4b, with 3 input cells, then from (8) the number of test patterns required will be 30 $<<$ 512.

Now, it is appropriate to ask and find out what should be the distribution of inputs to the cells in the second level. For example, the network with 9 inputs may be implemented in two levels with two cells in the second level one with 4 inputs and the other with 5 inputs. It is quite possible to have other configurations, for example some of the possible configurations for a 9 input network are 3,3,3; 4,5; 2,7 and etc. It is possible to have multi level netwoork with $s > 2$, then we would like to know what should be the optimal number of levels in the new design.

It can be easily shown that, if we restrict the number of levels in the new network to be $s = 2$, then from (8) one would obtain minimum number of test

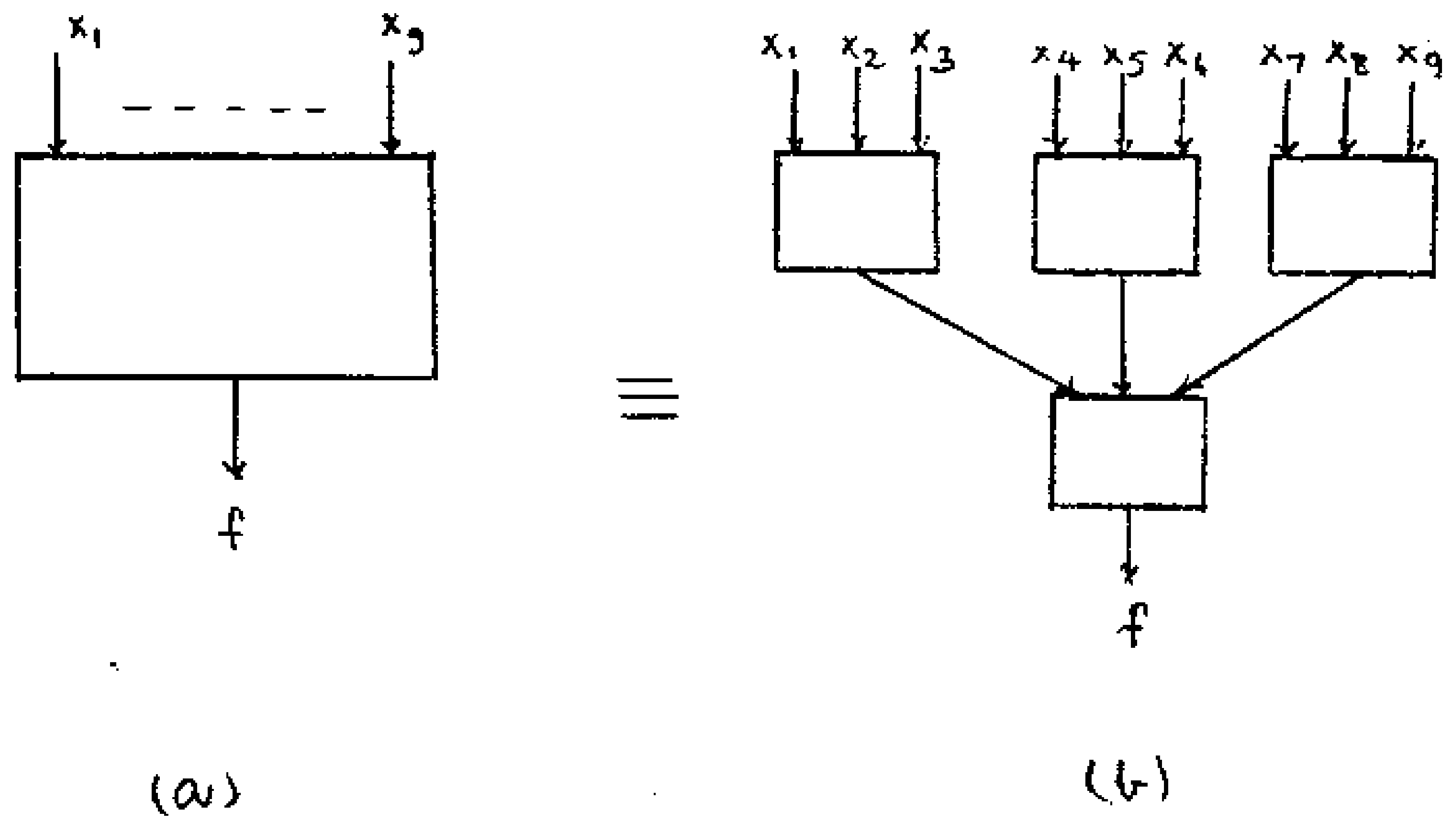

Figure 7: Different Implementations of a Network

patterns, if each cell in the second level has

$$m_i \leq \lfloor log_2 n \rfloor , \tag{10}$$

where $\lfloor K \rfloor$ is the integer part of K, and n is the number of input variables to a network. If $m_i > 2$, then once again (10) can be applied to the individual cells and each cell can be implemented as a two level network. This proceedure can be carried out, if possible, till every cell in a network is a two input cell. This can certainly be done if each cell in the original network is a single gate. In otherwords, the number of test patterns required will be minimized by implementing a network with two input cells (gates).

Now, let us return back to trees where each cell is a 2 input cell and the number of levels in the tree is s, then by (8)

$$N_s = 3.2^s - 2 = 3.n - 2.$$

Hence, the number of test patterns which would detect all multiple s-a-fs and single briging faults in a tree is upper bounded by $3.n - 2$. Hayes [1] has shown that the number of test patterns required to detect all single s-a-fs in a tree is upper bounded by $n + 1$.

If the gates used in a tree are limited to NAND, NOR, AND and OR gates only, then it can be shown that for a m-input gate all single and multiple s-a-fs can be detected by at most $m + 1$ test patterns. So, it is not necessary to apply all 2^m input vectors to every m-input gate. Then for a netwrk with s-levels, the equations (7) and (8) would become

$$N_s = m.N_{s-1} + (m+1) - 2, \quad N_1 = m + 1.$$

$$N_s = m^s + 2.m^{s-1} - 1 = n + 2.\frac{n}{m} - 2, \tag{11}$$

where n is the number oof input variables to a tree. If $m = 2$, then

$$N_s = 2.n - 1, \quad s > 1,$$

which is not very far from the single s-a-f test set.

If a network is constructed with NAND, AND, OR and NOR gates only, then equation (3) can be written as

$$|T| \leq \sum_{i=1}^{M} (m_i + 1) - 2 = \sum_{i=1}^{M} m_i + M - 2\,, \tag{12}$$

where M is the number of cells in a network. The first term $\sum_{i=1}^{M} m_i$ is nothing but the number of lines in a network. Hence, the total number of test patterns is linearly proportional to the number of lines a network.

Example 2 : *Consider a network with 16 outputs, 20 input variables and 50 product terms, where each product term is a function of 8 variables and each output O_i is a function of 16 product terms.*

The number of test pattens computed by (12) is 464.

REFERENCES:

[1]. Hayes, J.P., "On Realization of Boolean Functions Requiring a Minimal or Near Minimal Number of Tests," IEEE Trans. on Computers, Vol. C-20, Dec. 1971, pp. 1506-1513.

[2]. Gault, J.W., Robinson, J.P., and Reddy, S.M., "Multiple Fault Detection in Combinational Networks," IEEE Trans. on Computers, Vol. C-21, Jan. 1972, pp. 31-37.

[3]. Kohavi, I., and Kohavi, Z., "Detection of Multiple Faults in Combinational Logic Networks," IEEE Trans. on Computers, Vol. C-21, June 1972, pp. 556-568.

[4]. Yau, S.S., and Yang, S.C., "Multiple Fault Detection for Combinational Logic Circuits," IEEE Trans. on Computers, Vol. C-24, Mar. 1975, pp. 233-241.

[5]. Karpovsky, M., and Su, S.Y.H., "Detection and Location of Input and Feedback Bridging Faults among Input and Output lines," IEEE Trans. on Computers, Vol. C-29, June 1980, pp. 523-527.

[6]. Kohavi, Z., and Berger, I., "Fault Diagnosis in Combinational Tree Ntworks," IEEE Trans. on Computers, Vol. C-24, Dec. 1975, pp. 1161-1167.

[7]. Sellers, E.F., Hsiao, M.Y., and Bearson, L.W., "Analyzing Errors with Boolean Difference," IEEE Trans. on Computers, Vol. C-17, 1968, pp. 676-683.

[8]. Ibarra, O.H., and Sahni, S.K., "Polynomially Complete Fault Detection Problems," IEEE Trans. on Computers, Vol. C-24, Mar. 1975, pp. 242-249.