Science Bulletin of NCTU, Vol. VII, No. 1

MINIMUM TEST SETS FOR FAN-OUT FREE NETWORKS AND TWO-LEVEL NETWORKS

SHIN-MAAN LEE
College of Engineering, National Chiao Tung University

and

CHARLES A. HARLOW
Department of Electrical Engineering, University of Missouri-Columbia

(Received 5 December 1973)

Abstract—*Algorithms are presented for generating minimum multiple-fault detection test lets for fan-out free networks and nonredundant two-level networks. A test set is minimum in the sense that there is no set with a smaller number of test patterns that detects all the faults in the network.*

1. INTRODUCTION

This paper presents algorithms for generating minimum test sets of test patterns which detect all the stuck-at type faults in fan-out free networks and two-level networks.

In analyzing the faults in a network we investigate the faulty out-put functions that are transformed from the normal output function by fault or faults in the network. If a set of test patterns which can differentiate all the faulty output functions from the normal output function, then obviously this set of test patterns detects all the faults in the network. In Fig. 1 we denote the set of all faulty output functions and the normal output function of AND gate g_1 by g_{1F}: $(0, 1, \overline{A}, B, \overline{A}B)$. Similarly we have g_{3F}: $(0, 1, C, g_{1F}, C+g_{1F})$ and Z_F: $(0, 1, g_{3F}, \bar{g}_{2F}, g_{3F}\cdot\bar{g}_{2F})$. It should be noted that NOT gates are considered as lines in the analysis. Set Z_F is used for analysis only, it is not needed in generating the test sets.

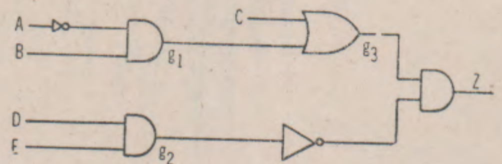

Fig. 1 Fan-out free network

2. MINIMUM TEST SETS GENERATION ALGORITHM FOR FAN-OUT FREE NETWORKS (MTFF)

The fan-out free networks considered in this section are with no input fan-out or internal fan-out. In general the output function of a fan-out free network Z can be written as $Z=g_iZ_1+Z_2$, where g_i is either an independent primary input or a primary gate, Z_1 and Z_2 are functions of primary gates and/or independent

primary inputs. A *primary gate* is a gate with all its inputs which are primary inputs. A primary input is called an *independent primary input* if is is not an input to a primary gate.

By expanding Z_1 we have $Z=g_i(m_1+......+m_k)+Z_2$, where $m_1,......, m_k$ are products of primary input variables. Later in the algorithm we will use the product $(m_1......m_k)$, which can be obtained without actually expanding Z_1. The product $(m_1......m_k)$ is obtained by simply multiplying all the literals which appear in Z_1.

The algorithm MTFF can now be stated.

Step 1. Express the output function Z in sum-of-product form in terms of primary gates and indepent primary inputs.

Step 2. For each g_i in Z generate all the $g_i^*(m......m_k)$'s; (a) if g_i is a product of variables, then the g_i^*'s are one variable less specified than g_i; (b) if g_i is a sum of variables or an independent primary input variable, then $g_i^*=1$. If $(g_i^*\ m_1......m_k) \geq (g_j^*\ m_t......m_v)$, $(g_i^*\ m_1......m_t)$ is discarded from the list.

Step 3. For all the $g_i^*(m_1......m_k)$'s in the list compute

$$T_m(g_i^*)=(g_i^*\ m_1......m_k)\bar{Z}.$$

Step 4. Find a minimum set $T(\bar{Z})$ which covers all the sets $T_m(g_i^*)$'s obtained in Step 3.

Step 5. Express $\bar{Z}$ in sum-of-product form in terms of primary gates and independent primary inputs. In general $\bar{Z}=g_jZ_3+Z_4=g_j(n_1+......+n_t)+Z_4$.

Step 6. For each g_j in $\bar{Z}$ generate all the $g_j^*(n_1......n_t)$'s which are obtained according to the same rules in Step 2. If $(g_j^*n_1......n_t) \geq (g_k^*n_u......n_v)$, $(g_j^*n_1......n_t)$ is discarded from the list.

Step 7. For all the $(g_j^*n_1......n_t)$'s in the list compute

$$S_m(g_j^*)=(g_j^*n_1......n_t)Z.$$

Step 8. Find a minimum set $T(Z)$ which covers all the sets $S_m(g_j^*)$'s obtained in Step 7.

Step 9. $T=T(\bar{Z})+T(Z)$ is a minimum test set.

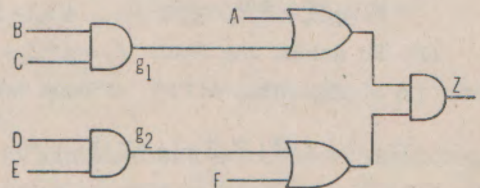

Fig. 2. Fan-out free network N_1

As an example, network N_1 in Fig. 2 is considered. Following the steps of MTFF we have

(1) $Z=AF+Ag_2+Fg_1+g_1g_2$,

(2) the list is: $(BDEF, CDEF, ABCD, ABCE)$,

(3) the $T_m(g_i^*)$'s are:

$(BDEF)\bar{Z}=\bar{A}B\bar{C}DEF=(23)$,

$(CDEF)\bar{Z}=\bar{A}\bar{B}CDEF=(15)$,

$(ABCD)\bar{Z}=ABCD\bar{E}\bar{F}=(60)$,

$(ABCE)\bar{Z}=ABC\bar{D}E\bar{F}=(58)$,

(4) $T(\bar{Z})=(15, 23, 58, 60)$,

(5) $\bar{Z}=\bar{A}\bar{g}_1+\bar{F}\bar{g}_2$,

(6) the list is: $(\overline{BC}, \bar{A}, \overline{DE}, \bar{F})$,

(7) the $S_m(g_j^*)$'s are:

$(\bar{B}\bar{C})Z=A\bar{B}\bar{C}F+A\bar{B}\bar{C}DE=(33, 35, 37, 38, 39)$,

$(\bar{D}\bar{E})Z=A\bar{D}\bar{E}F+BC\bar{D}\bar{E}F=(25, 33, 41, 49, 57)$,

$(\bar{A})Z=\bar{A}BCF+\bar{A}BCDE=(25, 27, 29, 30, 31)$,

$(\bar{F})Z=ADE\bar{F}+BCDE\bar{F}=(30, 38, 46, 54, 62)$,

(8) $T(Z)=(30, 33)$,

(9) $T=T(\bar{Z})+T(Z)=(15, 23, 58, 60, 30, 33)$ is a minimum test set for N_1.

Network N_1 was used by Kohavi and Kohavi[1] who showed that a test set of six test patterns can be generated by their method. However, it is not clear that their method will always generate a minimum test.

In the following we will (1) show that the test set derived for a fan-out free network detects all the stuck-at type faults, and (2) prove that the test set is minimum.

Before showing that all the faults can be detected we need to prove that the test set exists.

Lemma 1: For eyery g_t in Z, $T_m(g_t^*)\neq\phi$ and for every g_j in $\bar{Z}$, $S_m(g_j^*)\neq\phi$.

Proof: See APPENDIX I.

We can now show the next lemma.

Lemma 2: For a fan-out free network the test set derived by using algorithm MTFF detects all the stuck-at type faults.

Proof: See APPENDIX I.

To prove that the test set is minimum we need to show that (1) all the test patterns in T are necessary and (2) there is no set with smaller number of test patterns which detects all the faults in the network.

Let $Z=g_tZ_1+Z_2=g_1(m_1+......+m_k)+Z_2$ and

$\bar{Z}=\bar{g}_t\bar{Z}_2+\bar{Z}_1\bar{Z}_2=\bar{g}_t(n_1+......+n_t)+\bar{Z}_1(n_1+......+n_t)$.

(1) To detect the fault $Z_F=g_t^*Z_1+Z_2$, where g_t^* is determined by the rules in Step (2) of algorithm MTFF, at least one test pattern is needed from the set $T(g_t^*)$,

$$T(g_t^*)=Z_F\bar{Z}+\bar{Z}_FZ=(g_t^*Z_1)\bar{Z}=g_t^*(m_1+......+m_k)\bar{Z}.$$

In step 4. of algorithm MTFF, we select one test pattern from each set $T_m(g_t^*)=(g_t^*m_1......m_k)\bar{Z}$ which is a set smaller than the corresponding set $T(g_t^*)$. Therefore, all the test patterns in $T(\bar{Z})$ are necessary.

To detect the fault $\bar{Z}_F=\bar{g}_t^*\bar{Z}_2+\bar{Z}_1\bar{Z}_2$, at least one test pattern is needed from the set $S(\bar{g}_t^*)$,

$$S(\bar{g}_t^*)=\bar{Z}_FZ+Z_F\bar{Z}=(\bar{g}_t^*\bar{Z}_2)Z=\bar{g}_t^*(n_1+......+n_t)Z.$$

In Step 8. of algorithm MTFF, we select one test pattern from each set $S_m(\bar{g}_t^*)=(\bar{g}_t^*(n_1......n_t)Z$ which is a set smaller than the corresponding set $S(\bar{g}_t^*)$. Therefore, all the test patterns in $T(Z)$ are necessary.

(2) Since $T(\bar{Z})$ and $T(Z)$ are disjoint, to show that $T=T(\bar{Z})+T(Z)$ is minimum is equivalent to proving the next two lemmas.

Lemma 3: If $T(g_t^*)\cdot T(g_2^*)......T(g_n^*)\neq\phi$ for some $g_1, g_2......g_n$ in Z, then $T_m(g_1^*)\cdot$

$T_m(g_2^*)......T_m(g_n^*)\neq\phi$.

Proof: See APPENDIX II.

Lemma 4: If $S(g_1^*)\cdot S(g_2^*)......S(g_n^*)\neq\phi$ for some $g_1, g_2,......, g_n$ in $\bar{Z}$, then $S_m(g_1^*)\cdot S_m(g_2^*)......S_m(g_n^*)\neq\phi$.

Proof: Similar to the proof of Lemma 3.

From Lemmas 1, 2, 3, and 4 the following theorem is established.

Theorem 1: For any fan-out free network, the algorithm MTFF generates a minimum test set which detects all the stuck-at type faults in it.

For networks with NOR gates or NAND gates, algorihm MTFF can also be applied by treating a NOR gate as an OR gate followed by a NOT gate and a NAND gate as an AND gate followed by a NOT gate.

3. MINIMUM TEST SETS GENERATION ALGORITHMS FOR TWO-LEVEL NETWORKS

For nonredundant two-level AND-OR networks and two-level OR-AND networks, the same idea used in section II can be applied to generate the minimum test sets.

A. Minimum Test Sets for Two-Level AND-OR Networks

A two-level AND-OR network is shown in Fig. 3. (1) For the fault such that $Z_F=g_1+......+g_i^*+......+g_n$, at least one test pattern is needed from the set $T(g_i^*)=Z_F\bar{Z}+\bar{Z}_FZ=Z_F\bar{Z}=(g_i^*\bar{g}_i)\bar{Z}$, where g_i^*'s are one variable less specified than gi. Since the network is nonredundant, $T(g_i^*)\neq\phi$. (2) For the fault $g_i=0$, *i.e.*, $Z_F=g_1......+g_{i-1}+g_{i+1}......+g_n$, at least one test pattern is needed from the set $S(g_i^*)=Z_F\bar{Z}+\bar{Z}_FZ=g_i(\bar{g}_1......\bar{g}_{i-1}\bar{g}_{i+1}......\bar{g}_n)$. Since the network is nonredundant, $S(g_i^*)\neq\phi$.

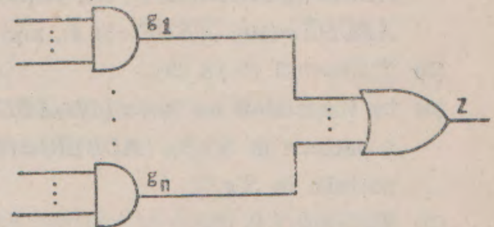

Fig 3. Two-level AND-OR network

It is obvious that the set $T=T(\bar{Z})+T(Z)$ is minimum and detects all the stuck-at faults in the network, where $T(\bar{Z})$ is the minimum set that covers all the $T(g_i^*)$'s and $T(Z)$ is the minimum set that covers all the $S(g_1^*)$'s.

The minimum test set generation algorithm can now be stated.

Step 1. For each g_i in Z generate all the $g_i^*\bar{g}_i$'s which are obtained by each time substituting one literal in g_i by its complement. If $g_i^*\bar{g}_i\geq g_j^*\bar{g}_j$, $g_i^*\bar{g}_i$ is discarded from the list.

Step 2. For all the $g_i^*\bar{g}_i$'s in the list compute

$$T(g_i^*)=(g_i^*\bar{g}_i)\bar{Z}.$$

Step 3. $T(\bar{Z})$ is the minimum set which covers all the sets obtained in Step 2.

Step 4. For each g_i in Z compute

$$S(g_i^*)=g_i(\bar{g}_1......\bar{g}_{i-1}\bar{g}_{i+1}......\bar{g}_n).$$

Actually only one arbitrary test pattern is needed from each $S(g_i^*)$, which can

be obtained by inspection: select one literal from each $\bar{g}_1, \ldots\ldots, \bar{g}_{t-1}, \bar{g}_{t+1}, \ldots\ldots, \bar{g}_n$ such that the product of g_t and the literals selected is not empty; if there is a variable which does not appear in the product, it can be put in the product in either form, complemental or not complemented.

Step 5. $T(Z)$ is the set consisting of one test pattern from each of the $S(g_i^*)$'s. $T(Z)$ is minimum due to that $S(g_i^*)\cdot S(g_j^*)=\phi$, for $i \neq j$.

Step 6. The minimum test set $T=T(\bar{Z})+T(Z)$.

As an example, Fig. 4 is a nonredundant network used by Kohavi and Kohavi[1] who showed the minimum test set consisting of nine test patterns. Following the steps outlined above we find that

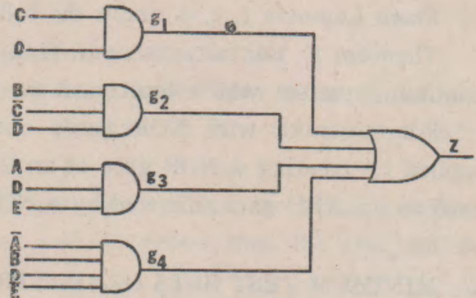

Fig. 4. An example of nonredundant AND-OR network

(1) the list is: $(\bar{B}\bar{C}\bar{D}, BC\bar{D}, B\bar{C}D, A\bar{D}E, A\bar{B}D\bar{E}, \bar{A}\bar{B}D\bar{E}, \bar{A}\bar{B}\bar{D}\bar{E}, \bar{A}\bar{B}DE)$,

(2) the $T(g_i^*)$'s are: $\bar{B}\bar{C}\bar{D}=(0,1,16,17)$, $BC\bar{D}=(12,13,28,29)$, $\bar{A}B\bar{C}D+B\bar{C}D\bar{E}=(10,11,26)$, $A\bar{B}\bar{D}E+AC\bar{D}E=(17,21,29)$, $A\bar{B}\bar{C}D\bar{E}=(18)$, $\bar{A}B\bar{C}D\bar{E}=(10)$, $\bar{A}\bar{B}\bar{D}\bar{E}=(0,4)$, and $\bar{A}\bar{B}\bar{C}DE=(3)$,

(3) $T(\bar{Z})=(0,3,10,18,29)$,

(4) by inspection we have $(CD)\bar{A}\bar{B}E=(7)$, a test pattern in $S(g_1^*)$, $(B\bar{C}\bar{D})\bar{A}E=(9)$ a pattern in $S(g_2^*)$, $(ADE)\bar{B}\bar{C}=(19)$, a pattern in $S(g_3^*)$, and $(\bar{A}\bar{B}D\bar{E})\bar{C}=(2)$ a pattern in $S(g_4^*)$,

(5) $T(Z)=(2,7,9,19)$,

(6) $T=T(\bar{Z})+T(Z)=(0,3,10,18,29,2,7,9,19)$ which consists of nine test patterns.

B. Minimum Test Sets for Two-Level OR-AND Networks

A two-level OR-AND network is shown in Fig. 5. The algorithm to be given here is similar to the algorithm for two-level AND-OR network, except that $\bar{Z}$ is used intead of Z, to generate the minimum test sets.

The algorithm for OR-AND networks is stated as follows.

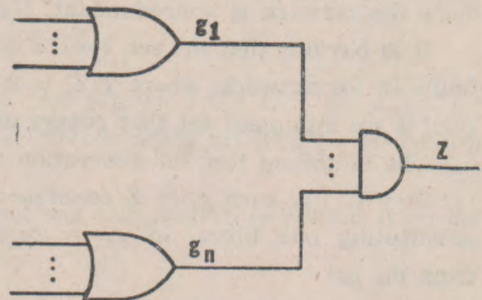

Fig. 5. Two-Level OR-AND network

Step 1. For each $\bar{g}_t$ in Z generate all the $\bar{g}_i^*g_i$'s which are obtained by each time substituting one literal in $\bar{g}_t$ by its complement. If $\bar{g}_i^*g_i \geq \bar{g}_j^*{}_j, \bar{g}_tg_t$ is discarded from the list.

Step 2. For all the $\bar{g}_tg_i$'s in the list compute

$$T(\bar{g}_i^*)=(\bar{g}_i^*g_i)Z.$$

Step 3. $T(Z)$ is the minimum set which covers all the sets obtained in Step 2.

Step 4. For each $\bar{g}_i$ in $\bar{Z}$ compute

$$S(\bar{g}_i^*)=\bar{g}_i(g_1\ldots\ldots g_{i-1}g_{i+1}\ldots\ldots g_n).$$

Similar to Step 4. of the algorithm for AND-OR networks, $S(\bar{g}_i^*)$ can be obtained by inspection.

Step 5. $T(\bar{Z})$ is the set consisting of one arbitrary test pattern from each of the $S(\bar{g}_i^*)$'s. $T(Z)$ is minimum due to that $S(\bar{g}_i^*)\cdot S(\bar{g}_j^*)=\phi$, for $i\neq j$.

Step 6. The minimum test set $T=T(Z)+T(\bar{Z})$.

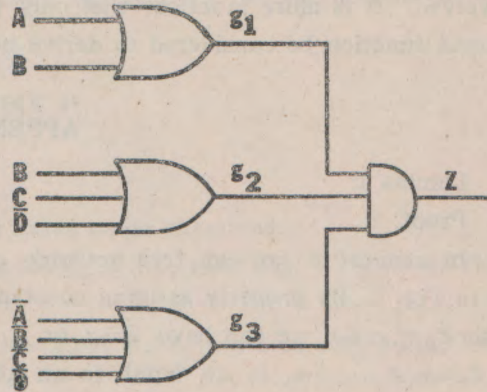

Fig. 6. An example of nonredundant OR-AND network

As an example, Fig. 6 is a nonredundant network used by Kohavi and Kohavi[1] who showed that the minimum number of test patterns needed is eight. According to the algorithm outlined we have following results:

(1) the list is: $(\bar{B}\bar{C}\bar{D}, \bar{A}BCD, A\bar{B}CD, AB\bar{C}D, AB\bar{C}D, ABC\bar{D})$,

(2) the $T(\bar{g}_i^*)$'s are: $A\bar{B}\bar{C}\bar{D}=(8)$, $\bar{A}BCD=(7)$, $A\bar{B}CD=(11)$, $AB\bar{C}D=(13)$, and $ABC\bar{D}=(14)$,

(3) $T(Z)=(7, 8, 11, 13, 14)$,

(4) by inspection we have $(\bar{A}\bar{B})C\bar{D}=(2)$ a test pattern in $S(\bar{g}_1^*)$, $(\bar{B}\bar{C}D)A=(9)$ a test pattern in $S(\bar{g}_2^*)$, and $(ABCD)=(15)$ a test pattern in $S(\bar{g}_3^*)$,

(5) $T(\bar{Z})=(2, 9, 15)$,

(6) $T=T(Z)+T(\bar{Z})=(7, 8, 11, 13, 14, 2, 9, 15)$ which consists of eight test patterns.

Compared to Kohavi and Kohavi's method[1], the algorithms presented here for AND-OR or OR-AND networks do not needed any maps and can handle networks with large number of variables as well.

4. DISCUSSION

For a fan-out free network N it is well known[2] that there exists a minimum single fault detection test set which also detects all the multiple faults in N. A simple algorithm MTFF is presented to generate the minimum test sets for fan-out free networks. In addition, algorithm MTFF needs fewer computational efforts than the generation of single fault test sets in the sense that $T_m(g_i^*)=(g_i^*m_1\ldots\ldots m_k)\bar{Z}$ and $S_m(g_j^*)=(g_j^*n_1\ldots\ldots n_k)Z$ are computed instead of $T(g_i^*)=g_i^*(m_1+\ldots\ldots+m_k)\bar{Z}$ and $S(g_j^*)=g_j^*(n_1+\ldots\ldots+n_t)Z$. Also, the sets $T_m(g_i^*)$ and $S_m(g_j^*)$ are smaller than the corresponding set $T(g_i^*)$ and $S(g_j^*)$, it is easier to find the covering sets for $T_m(g_i^*)$'s and $S_m(g_j^*)$'s, respectively.

For nonredundant two-level networks, Kohavi and Kohavi's method[1] uses maps in generating the minimum test sets, but it has difficulties in handling networks with large number of variables. The algorithm presented in section 3 can generate

minimum test sets for networks with large number of variables in a straightforward manner.

For networks in general, if all the faulty output functions are considered in order to generate the minimum test sets, then large computational tasks will be involved. It is more practical that only part of the information from each faulty output function be considered to derive near minimum test sets for the networks[3].

APPENDIX I

Lemma 1.

Proof:

In general a fan-out free network can be shown as in Fig. 7. By properly assiging constant values 0 or 1 to $Z_1, \ldots\ldots, Z_n$, we can have $Z=g_i$ or $\bar{g}_i$. Particularly if $Z_j=m_1+\ldots\ldots+m_t$ is an input to an AND gate, then assign $m_1=\ldots\ldots=m_t=1$; if Z_j is an input to an OR gate, then assign $m_1=\ldots\ldots=m_t=0$.

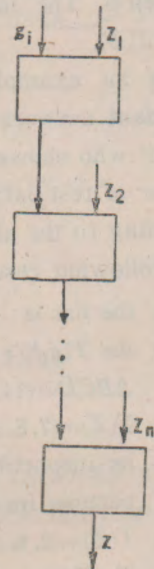

Fig. 7. Fan-out free network

Let $Z=g_iZ_1+Z_2=g_i(m_1+\ldots\ldots+m_k)+n_1\ldots\ldots n_v$.

(1) Suppose $(g_i{*}m_1)\bar{Z}=\phi$, *i.e.*, $g_i{*}\bar{g}_im_1(\bar{n}_1\ldots\ldots\bar{n}_k\ldots\ldots\bar{n}_v)=\phi$, then for some t, $m_1\bar{n}_t=\phi$ which implies that $m_1\leq n_t$. Since $m_1\leq n_t$, now if by assigning $m_1=\ldots\ldots=m_k=1$ in order to obtain $Z=g_i$ we will have $Z=1$ instead of $Z=g_i$, which contradicts that the network is fan-out free. Thus, $(g_i{*}m_1)\bar{Z}\neq\phi$.

(2) Suppose $(g_i{*}m_1m_2)\bar{Z}=\phi$, then for some t we will have $m_1m_2\bar{n}_t=\phi$, which implies that $m_1m_2\leq n_t$. By using the same argument as in (1) we can deduce that $(g_i{*}m_1m_2)\bar{Z}\neq\phi$.

(3) Obviously for a fan-out free network $(g_i{*}m_1\ldots\ldots m_k)\bar{Z}\neq\phi$.

Similarly, if $\bar{Z}=g_i(m_1+\ldots\ldots+m_k)+Z_2$, then $(g_i{*}m_1\ldots\ldots m_k)Z\neq\phi$.

Lemma 2.

Proof: Let $Z=g_iZ_1+Z_2=g_i(m_1+\ldots\ldots+m_k)+Z_2$ and $\bar{Z}=\bar{g}_i\bar{Z}_2+\bar{Z}_1\bar{Z}_2=\bar{g}_i(n_1+\ldots\ldots+n_t)+\bar{Z}_1(n_1+\ldots\ldots+n_t)$.

(1) For g_i is an AND gate, or the complement of an OR gate, or an independent primary input variable, if one test pattern is selected from $T_m(g_i{*})=(g_i{*}m_1\ldots\ldots m_k)\bar{Z}$, then the test pattern selected detects all the faults involving a fault or faults in g_i, except those faults which are equivalent to all those faults which are equivalent to all those faults with $g_i=0$. All the faults with $g_i=0$ can be detected by any test pattern in $S_m(\bar{g}_i{*})=(\bar{g}_i{*}n_1\ldots\ldots n_t)Z=(n_1\ldots\ldots n_t)Z$.

(2) For g_i is an OR gate or the complement of an AND gate, if one test pattern is selected from $S_m(\bar{g}_i{*})=(\bar{g}_i{*}n_1\ldots\ldots n_t)Z$, then the test pattern selected detects all the faults involving a fault or faults in g_i, except those faults which are equivalent to all the faults with $g_i=1$ and $Z_2=0$. All the faults with $g_i=1$ and $Z_2=0$ can be detected by any test pattern selected from $T_m(g_i{*})=(g_i{*}m_1\ldots\ldots m_k)\bar{Z}=(m_1\ldots\ldots m_k)\bar{Z}$.

(3) For all those faults such that $Z_F=1$, it can be detected by any test pattern in $T(\bar{Z})$ and for all those faults such that $Z_F=0$, it can be detected by any test pattern in $T(Z)$.

In algorithm MTFF we compute $T_m(g_i^*)$ and $S_m(g_i^*)$ for all primary gates and independent primary input variables; therefore the test set derived detects all the faults.

APPENDIX II

Lemma 3.

Proof:

Suppose $T(g_i^*)\cdot T(g_2^*)\neq\phi$, there are four cases to be examined.

(1) If the output function is in the form $Z=g_1g_2Z_1+\bar{Z}_2$ then $T(g_1^*)\cdot T(g_2^*)=\phi$. This case therefore need not be considered.

(2) If the output function is in the form $Z=g_1g_2Z_1+g_1Z_2+Z_3$, then $T(g_1^*)\cdot T(g_2^*)=\phi$. So this case need not to be considered.

(3) If $Z=g_1g_2Z_1+g_1Z_2+g_2Z_3+Z_4$ then $T(g_1^*)\cdot T(g_2^*)=g_1^*g_2^*(Z_2Z_3)\bar{Z}_4$. For a fan-out free network the relation $Z_2Z_3\leq Z_4$ holds, therefore $T(g_1^*)\cdot T(g_2^*)=\phi$. This case consequently need not to be considered. Note: We will show that $Z_2Z_3\leq Z_4$ later in the back of the proof.

(4) If the output function is in the form $Z=g_1Z_1+g_2Z_2+Z_3=g_1(m_1+\ldots\ldots+m_t)+g_2(n_1+\ldots\ldots+n_r)+(k_1+\ldots\ldots+k_2)$, this is the only case where $T(g_1^*)\cdot T(g_2^*)\neq\phi$, Suppose $T_m(g_1^*)\cdot T_m(g_1^*)=\phi$, *i.e.*, $(g_1^*g_2^*)\ (\bar{g}_1\bar{g}_2)\ (m_1\ldots\ldots m_k.\, n_1\ldots\ldots n_r)\bar{k}_1\ldots\ldots\bar{k}_s=\phi$ which implies that for some i, $(m_1\ldots\ldots m_k.\, n_1\ldots\ldots n_r)\bar{k}_t=\phi$, which is equivalent to $(m_1\ldots\ldots m_k.\, n_1\ldots\ldots n_r)\leq k_t$. Similar to the argument used in the proof of Lemma 1, if we assign $m_1=\ldots\ldots=m_t=1$ and $n_1=\ldots\ldots=n_r=1$ in order to obtain $Z=g_1+g_2$, we will have $Z=1$ instead. This contradicts that Z is the output function of a fan-out free network. Thus, $T_m(g_1^*)\cdot T_m(g_2^*)\neq\phi$.

Suppose $T(g_1^*)\cdot T(g_2^*)\cdot T(g_3^*)\neq\phi$, then from (1), (2), (3) and (4) above, we know that the output function is in the form $Z=g_1Z_1+g_2Z_2+g_3Z_3+Z_4$. Similar to the argument used in case (4) above, we can deduce that $T_m(g_1^*)\cdot T_m(g_2^*)\cdot T_m(g_3^*)\neq\phi$.

Extend to n elements, if $T(g_1^*)\ldots\ldots T(g_n^*)\neq\phi$ then $T_m(g_1^*)\ldots\ldots T_m(g_n^*)\neq\phi$. The lemma is proved.

Next we will show that if the output function is in the form $Z=g_1g_2Z_1+g_1Z_2+g_2Z_3+Z_4$ then $Z_2Z_3\leq Z_4$.

(1) Suppose g_1 and g_2 first meet at an AND gate as indicated in Fig. 8(a), we have $Z_t=g_1g_2Z_aZ_c+g_1Z_aZ_d+g_2Z_bZ_c+Z_bZ_d$ which is in the form $Z_t=g_1g_2Z_1+g_1Z_2+g_2Z_3+Z_4$ and with $Z_2Z_3\leq Z_4$. If Z_t is not the output function and goes through several AND gates or OR gates but not any NOT gates then the function Z_u will still be in the form $Z_u=g_1Z_1+g_2Z_2+g_3Z_3+Z_4$ with $Z_2Z_3\leq Z_4$. Once Z_4 enters a NOT gate, then the resultant function Z_v is in the form $Z_v=\bar{g}_1Z_m+\bar{g}_2Z_n+Z_l$.

(2) Suppose g_1 and g_2 first meet at an OR gate as indicated in Fig. 8(b), we have $Z_t=g_1Z_a+g_2Z_c+Z_b+Z_d$ which is in the form $Z_t=g_1Z_m+g_2Z_n+Z_l$. If Z_t goes through several AND gates or OR gates but not any NOT gates, then the resultant function

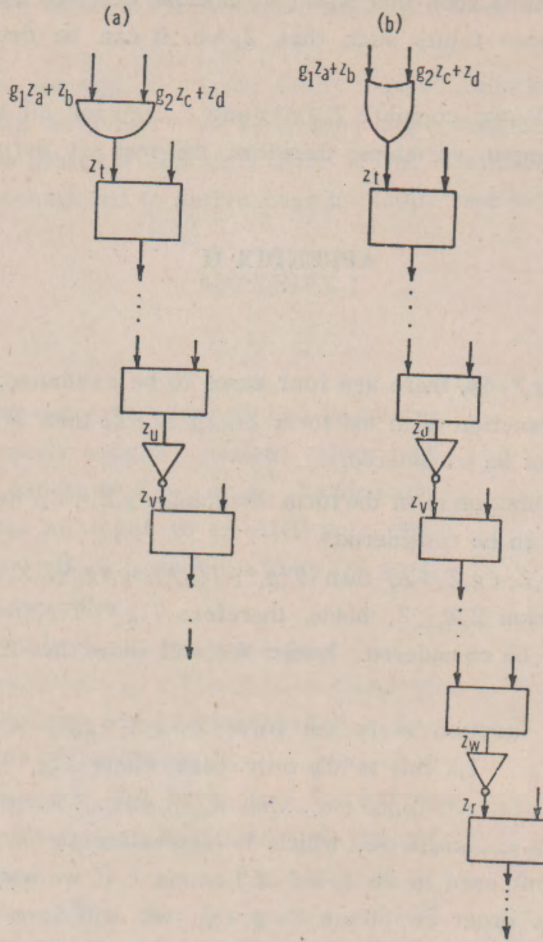

Fig. 8. (a) g_1 and g_2 meet at an AND gate
(b) g_1 and g_2 meet at an OR gate.

Z_u is still in the form $Z_u = g_1 Z_m + g_2 Z_n + Z_l$. Once Z_u enters a NOT gate, then the resultant function is $Z_v = \bar{g}_1 \bar{g}_2 \bar{Z}_l + \bar{g}_1 \bar{Z}_n \bar{Z}_l + \bar{g}_2 \bar{Z}_m \bar{Z}_l + \bar{Z}_m \bar{Z}_n \bar{Z}_l$, which is in the form $Z_v = \bar{g}_1 \bar{g}_2 Z_1 + \bar{g}_1 Z_2 + \bar{g}_2 Z_3 + Z_4$ with $Z_2 Z_3 \leq Z_4$. If Z_v goes through several AND gates or OR gates but not any NOT gates, the resultant function Z_w remains in the form $Z_w = \bar{g}_1 \bar{g}_2 Z_1 + \bar{g}_1 Z_2 + \bar{g}_2 Z_3 + Z_4$ with $Z_2 Z_3 \leq Z_4$. Once Z_w enters a NOT gate then the resultant function will be in the form $Z_r = g_1 Z_m + g_2 Z_n + Z_l$. (1) and (2) together show that if the output function of a fan-out free network is in the form $Z = g_1 g_2 Z_1 + g_1 Z_2 + g_2 Z_3 + Z_4$, then the relation $Z_2 Z_3 \leq Z_4$ holds.

REFERENCES

1. I. Kohavi and Z. Kohavi, "Detection of multiple faults in combinational logic networks," IEEE Trans. Comput., Vol. C-21, pp. 556-568, June 1972.
2. J.P. Hayes, "A NAND model for fault diagnosis in combinational logic networks," IEEE Trans. Comput., Vol. C-20, pp. 1496-1506, Dec. 1971.
3. D.C. Bossen and S.J. Hong, "Cause-effect analysis for multiple fault detection in combinational network," IEEE Trans. Comput., Vol. C-20, pp. 1252-1257, Nov. 1971.
4. F.F. Sellers, Jr., M.Y. Hsiao, and L.W. Bearnson, "Analyzing errors with the Boolean difference," IEEE Trans. Comput., Vol. C-17, pp. 676-683, July 1968.