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 sets 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 $g_{1,F}$: (0, 1, $\bar{A}$, $B$, $\bar{A}B$). Similarly we have $g_{1,F}$: (0, 1, C, $g_{1,F}$, C+$g_{1,F}$) and $Z_F$: (0, 1, $g_{1,F}$, $\bar{g}_{1,F}$, $g_{1,F}$$\bar{g}_{1,F}$). 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.

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
\[
Z = g_1(m_1 + \ldots + m_k) + Z_2,
\]
where \( m_1, \ldots, m_k \) are products of primary input variables. Later in the algorithm we will use the product \( (m_1, \ldots, m_k) \), which can be obtained without actually expanding \( Z_1 \). The product \( (m_1, \ldots, m_k) \) 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 \( g_t \) in \( Z \) generate all the \( g_t^k(m_1, \ldots, m_k) \)'s; (a) if \( g_t \) is a product of variables, then the \( g_t^k \)'s are one variable less specified than \( g_t \); (b) if \( g_t \) is a sum of variables or an independent primary input variable, then \( g_t^k = 1 \). If \( (g_t^k m_1, \ldots, m_k) \geq (g_t^k m_1, \ldots, m_t) \), \( (g_t^k m_1, \ldots, m_t) \) is discarded from the list.

Step 3. For all the \( g_t^k(m_1, \ldots, m_k) \)'s in the list compute
\[
T_w(g_t^k) = (g_t^k m_1, \ldots, m_k)Z.
\]

Step 4. Find a minimum set \( T(\bar{Z}) \) which covers all the sets \( T_w(g_t^k) \)'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
\[
\bar{Z} = g_jZ_2 + Z_4 = g_j(n_1 + \ldots + n_t) + Z_4.
\]

Step 6. For each \( g_j \) in \( Z \) generate all the \( g_j^k(n_1, \ldots, n_t) \)'s which are obtained according to the same rules in Step 2. If \( (g_j^k n_1, \ldots, n_t) \geq (g_j^k n_1, \ldots, n_t) \), \( (g_j^k n_1, \ldots, n_t) \) is discarded from the list.

Step 7. For all the \( (g_j^k n_1, \ldots, n_t) \)'s in the list compute
\[
S_w(g_j^k) = (g_j^k n_1, \ldots, n_t)Z.
\]

Step 8. Find a minimum set \( T(Z) \) which covers all the sets \( S_w(g_j^k) \)'s obtained in Step 7.

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

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

1. \( Z = AF + Ag + Fg + g \).
2. the list is: \( BDEFG, CDEFG, ABCD, ABCE \).
3. the \( T_w(g_t^k) \)'s are:
   \[
   \begin{align*}
   (BDEFG)\bar{Z} &= ABCE = (23), \\
   (CDEFG)\bar{Z} &= ABCDE = (15), \\
   (ABCD)\bar{Z} &= ABCDEF = (60), \\
   (ABCE)\bar{Z} &= ABCDEF = (58),
   \end{align*}
   \]
4. \( T(\bar{Z}) = (15, 23, 58, 60) \).
5. \( \bar{Z} = \overline{A}g + \overline{F}g \).
6. the list is: \( BC, A, DE, F \).

Fig. 2. Fan-out free network \( N_1 \).
(7) the $S_m(g_i^*)$s are:
\[
(\overline{BC}) \overline{Z} = A\overline{BC}F + A\overline{BCDE} = (33, 35, 37, 38, 39),
\]
\[
(\overline{DE}) \overline{Z} = A\overline{DE}F + B\overline{CDEF} = (25, 33, 41, 49, 57),
\]
\[
(\overline{A}) \overline{Z} = A\overline{BC}F + A\overline{BCDE} = (25, 27, 29, 30, 31),
\]
\[
(\overline{F}) \overline{Z} = A\overline{DE}F + B\overline{CDEF} = (30, 38, 46, 54, 62).
\]

(8) $T(\overline{Z}) = (30, 33)$,

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

Network $N_1$ was used by Kohavi and Kohavi 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 every $g_i$ in $Z$, $T_m(g_i^*) \neq \phi$ and for every $g_i$ in $\overline{Z}$, $S_m(g_i^*) \neq \phi$.

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 $Z = g_1Z_1 + Z_2 = g_1(m_1 + \ldots + m_b) + Z_2$ and $\overline{Z} = \overline{g_1}Z_1 + \overline{Z}_2 = \overline{g_1}(n_1 + \ldots + n_c) + \overline{Z}_2(n_1 + \ldots + n_c)$.

(1) To detect the fault $Z_p = g_i^*Z_1 + Z_{p_a}$, where $g_i^*$ is determined by the rules in Step (2) of algorithm MTFF, at least one test pattern is needed from the set $T(g_i^*)$.

\[
T(g_i^*) = Z_pZ + \overline{Z}_pZ = (g_i^*Z_1)Z = g_i^*(m_1 + \ldots + m_b)Z.
\]

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

To detect the fault $\overline{Z}_p = \overline{g_i^*Z_1} + \overline{Z}_2$, at least one test pattern is needed from the set $S(\overline{g_i^*})$.

\[
S(\overline{g_i^*}) = \overline{Z}_pZ + Z_p\overline{Z} = (\overline{g_i^*Z_1})Z = \overline{g_i^*}(n_1 + \ldots + n_c)Z.
\]

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

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

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

[Page 28]
$T_m(g_1^*)...T_m(g_n^*) \neq \emptyset$

Proof: See APPENDIX II.

Lemma 4: If $S(g_1^*), S(g_2^*), ..., S(g_n^*) \neq \emptyset$ for some $g_1, g_2, ..., g_n$ in $Z$, then $S_m(g_1^*), S_m(g_2^*), ..., S_m(g_n^*) \neq \emptyset$.

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, algorithm 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 $Z_p = g_1 + ... + g_i^* + ... + g_n$, at least one test pattern is needed from the set $T(g_i^*) = Z_pZ + Z_pZ = Z_pZ = (g_i^*g_i)Z$, where $g_i^*$'s are one variable less specified than $g_i$. Since the network is nonredundant, $T(g_i^*) \neq \emptyset$. (2) For the fault $g_i = 0$, i.e., $Z_p = g_1 + ... + g_i + ... + g_n$, at least one test pattern is needed from the set $S(g_i^*) = Z_pZ + Z_pZ = g_iZ = g_1Z = ...Z_{i-1}Z_iZ_{i+1}...Z_n$. Since the network is nonredundant, $S(g_i^*) \neq \emptyset$.

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

The minimum test set generation algorithm can now be stated.

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

Step 2. For all the $g_i^*g_i$'s in the list compute

$T(g_i^*) = (g_i^*g_i)Z$.

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

Step 4. For each $g_i$ in $Z$ compute

$S(g_i^*) = g_iZ = ...Z_{i-1}Z_iZ_{i+1}...Z_n$.

Actually only one arbitrary test pattern is needed from each $S(g_i^*)$, which can
be obtained by inspection: select one literal from each \(\overline{g}_1, \ldots, \overline{g}_{r-1}, \overline{g}_{r+1}, \ldots, \overline{g}_n\) such that the product of \(g_i\) 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 \(S(g_i^*) = S(g_i) = g_i\) for \(i \neq j\).

Step 6. The minimum test set \(T = T(Z) + T(Z)\).

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

1. The list is: \((BCD, BCD, BCD, ADE, ABDE, ABDE, ABDE, ABDE)\),
2. The \(T(g_i^*)\)'s are: \(BCD = (0, 1, 16, 17), BCD = (12, 13, 28, 29), ABCD + BCDE = (10, 11, 26), ABDE + ACDE = (17, 21, 29), ABCD = (18), ABCDE = (10), ABDE = (0, 4), and ABCDE = (3),\)
3. \(T(Z) = (0, 3, 10, 18, 29),\)
4. By inspection we have \((CD)\overline{ABE} = (7), a test pattern in \(S(g_i^*), (BCD)\overline{AE} = (9), a pattern in \(S(g_i^*), (ADE)\overline{BC} = (19), a pattern in \(S(g_i^*), and (ABDE)\overline{C} = (2) a pattern in \(S(g_i^*)\),\)
5. \(T(Z) = (2, 7, 9, 19),\)
6. \(T = T(Z) + T(Z) = (0, 3, 10, 18, 29, 2, 7, 9, 19)\) 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 \(Z\) is used in stead of \(Z\), to generate the minimum test sets.

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

Step 1. For each \(g_i\) in \(Z\) generate all the \(g_i^*g_i\)'s which are obtained by each time substituting one literal in \(g_i\) by its complement. If \(g_i^*g_i = g_j^*g_j, g_i\) is discarded from the list.

Step 2. For all the \(g_i^*g_i\)'s in the list compute \(T(g_i^*) = (g_i^*g_i)Z\).

Step 3. \(T(Z)\) is the minimum set which covers all the sets obtained in Step 2.
Step 4. For each $g_i$ in $Z$ compute

$$S(g_i) = -g_i(g_1, \ldots, g_i, \ldots, g_n).$$

Similar to Step 4. of the algorithm for AND-OR networks, $S(g_i)$ can be obtained by inspection.

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

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

As an example, Fig. 6 is a non-redundant network used by Kohavi and Kohavi 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: $(BCD, A\bar{B}CD, A\bar{B}CD, AB\bar{C}D, AB\bar{C}D, AB\bar{C}D),$
2. the $T(g_i)$'s are: $ABCD = (8), \bar{A}BCD = (7), A\bar{B}CD = (11), AB\bar{C}D = (13),$ and $ABC\bar{D} = (14),$
3. $T(Z) = (7, 8, 11, 13, 14),$
4. by inspection we have $(\bar{A}B)CD = (2)$ a test pattern in $S(g_1)$, $(B\bar{C}D)A = (9)$ a test pattern in $S(g_3)$, and $(ABCD) = (15)$ a test pattern in $S(g_4)$,
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, 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 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 $T_m(g_i) = (g_i, m_1, \ldots, m_e)Z$ and $S_m(g_i) = (g_i, n_1, \ldots, n_e)Z$ are computed instead of $T(g_i) = g_i(m_1, \ldots, m_e)Z$ and $S(g_i) = g_i(n_1, \ldots, n_e)Z$. Also, the sets $T_m(g_i)$ and $S_m(g_i)$ are smaller than the corresponding set $T(g_i)$ and $S(g_i)$, it is easier to find the covering sets for $T_m(g_i)'s$ and $S_m(g_i)'s$, respectively.

For nonredundant two-level networks, Kohavi and Kohavi's method 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.

**APPENDIX I**

Lemma 1.

Proof:

In general a fan-out free network can be shown as in Fig. 7. By properly assigning constant values 0 or 1 to \( Z, \ldots, Z_n \), we can have \( Z=g_t \) or \( \bar{g}_t \). Particularly if \( Z_i=m_1+\ldots+m_k \) is an input to an AND gate, then assign \( m_i=\ldots=m_k=1 \); if \( Z_j \) is an input to an OR gate, then assign \( m_i=\ldots=m_k=0 \).

Let \( Z=g_tZ_i+Z_j=g_t(m_1+\ldots+m_k)+n_1,\ldots,n_t \).

1) Suppose \( (g_t\bar{g}_t)^nZ=\phi \), i.e., \( g_t\bar{g}_t^n(m_1,\ldots,m_k,\ldots,n_t)=\phi \), then for some \( t \), \( m_tn_t=\phi \) which implies that \( m_t\leq n_t \). Since \( m_t\leq n_t \), now if by assigning \( m_i=\ldots=m_k=1 \) in order to obtain \( Z=g_t \) we will have \( Z=1 \) instead of \( Z=g_t \), which contradicts that the network is fan-out free. Thus, \( (g_t\bar{g}_t)^nZ=\phi \).

2) Suppose \( (g_t\bar{g}_t)^n(m_t,\ldots,m_t)^nZ=\phi \), then for some \( t \) we will have \( m_tn_t=\phi \), which implies that \( m_t\leq n_t \). By using the same argument as in (1) we can deduce that \( (g_t\bar{g}_t)^nZ=\phi \).

3) Obviously for a fan-out free network \( (g_t\bar{g}_t)^nZ=\phi \).

Similarly, if \( Z=g_t(m_1+\ldots+m_k)+Z_j \), then \( (g_t\bar{g}_t)^nZ=\phi \).

Lemma 2.

Proof: Let \( Z=g_tZ_i+Z_j=g_t(m_1+\ldots+m_k)+Z_j \) and \( Z=\bar{g}_tZ_i+Z_j=\bar{g}_t(m_1+\ldots+n_t)+Z_j(n_1+\ldots+n_t) \).

1) For \( g_t \) is an AND gate, or the complement of an OR gate, or an independent primary input variable, if one test pattern is selected from \( T_{\infty}(\bar{g}_t^n)=(g_t\bar{g}_t)^n(n_1,\ldots,n_t)Z=(n_1,\ldots,n_t)Z \), then the test pattern selected detects all the faults involving a fault or faults in \( g_t \), except those faults which are equivalent to all those faults which are equivalent to all those faults with \( g_t=0 \). All the faults with \( g_t=0 \) can be detected by any test pattern in \( S_{\infty}(\bar{g}_t^n)=(g_t\bar{g}_t^n(n_1,\ldots,n_t)Z=(n_1,\ldots,n_t)Z) \).

2) For \( g_t \) is an OR gate or the complement of an AND gate, if one test pattern is selected from \( S_{\infty}(\bar{g}_t^n)=(g_t\bar{g}_t^n(n_1,\ldots,n_t)Z \), then the test pattern selected detects all the faults involving a fault or faults in \( g_t \), except those faults which are equivalent to all the faults with \( g_t=1 \) and \( Z_j=0 \). All the faults with \( g_t=1 \) and \( Z_j=0 \) can be detected by any test pattern selected from \( T_{\infty}(\bar{g}_t^n)=(g_t\bar{g}_t^n(m_1,\ldots,m_k)Z=(m_1,\ldots,m_k)Z \).
(3) For all those faults such that \(Z_F = 1\), it can be detected by any test pattern in \(T(Z)\) 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 \(T_w(g_i^*)\) and \(S_w(g_i^*)\) for all primary gates and independent primary input variables; therefore the test set derived detects all the faults.

**APPENDIX II**

Lemma 3.

**Proof:**

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

(1) If the output function is in the form \(Z = g_i^* Z_1 + \overline{Z_2} \) then \(T(g_i^*) \cdot T(g_j^*) = \phi\). This case therefore need not be considered.

(2) If the output function is in the form \(Z = g_i^* Z_1 + g_j^* Z_2 + Z_3\), then \(T(g_i^*) \cdot T(g_j^*) = \phi\). So this case need not to be considered.

(3) If \(Z = g_i^* Z_1 + g_j^* Z_2 + g_k^* Z_3 + Z_4\) then \(T(g_i^*) \cdot T(g_j^*) = g_k^* \). For a fan-out free network the relation \(Z_1 \leq Z_4\) holds, therefore \(T(g_i^*) \cdot T(g_j^*) = \phi\). This case consequently need not to be considered. Note: We will show that \(Z_i Z_1 \leq Z_4\) later in the back of the proof.

(4) If the output function is in the form \(Z = g_i^* Z_1 + g_j^* Z_2 + Z_3 = g_i^* Z_1 + g_j^* Z_2 + Z_3 + Z_4\), this is the only case where \(T(g_i^*) \cdot T(g_j^*) = \phi\). Suppose \(T_w(g_i^*) \cdot T_w(g_j^*) = \phi\), i.e., \((g_i^* g_j^*) (g_1^* Z_1 \cdot \overline{Z_2}) (m_1 \ldots m_k n_1 \ldots n_r) \neq \phi\) which implies that for some \(i_s, (m_1 \ldots m_k n_1 \ldots n_r) \neq \phi\), which is equivalent to \((m_1 \ldots m_k n_1 \ldots n_r) \leq k_s\). Similar to the argument used in the proof of Lemma 1, if we assign \(m_1 = \ldots = m_k = 1\) and \(n_1 = \ldots = n_r = 1\) in order to obtain \(Z = g_i^* + g_j^*\), we will have \(Z = 1\) instead. This contradicts that \(Z_i \leq Z_4\) is the output function of a fan-out free network. Thus, \(T_w(g_i^*) \cdot T_w(g_j^*) = \phi\).

Suppose \(T(g_i^*) \cdot T(g_j^*) \cdot T(g_k^*) = \phi\), then from (1), (2), (3) and (4) above, we know that the output function is in the form \(Z = g_i^* Z_1 + g_j^* Z_2 + g_k^* Z_3 + Z_4\). Similar to the argument used in case (4) above, we can deduce that \(T_w(g_i^*) \cdot T_w(g_j^*) \cdot T_w(g_k^*) = \phi\).

Extend to \(n\) elements, if \(T(g_i^*) \ldots T(g_n^*) = \phi\) then \(T_w(g_i^*) \ldots T_w(g_n^*) = \phi\). The lemma is proved.

Next we will show that if the output function is in the form \(Z = g_i^* Z_1 + g_j^* Z_2 + g_k^* Z_3 + Z_4\), then \(Z_i Z_1 \leq Z_4\).

(1) Suppose \(g_i^*\) and \(g_j^*\) first meet at an AND gate as indicated in Fig. 8(a), we have \(Z_i = g_i^* Z_1 + g_j^* Z_2 + g_k^* Z_3 + Z_4\), which is in the form \(Z_i = g_i^* Z_1 + g_j^* Z_2 + g_k^* Z_3 + Z_4\) and with \(Z_i Z_1 \leq Z_4\). If \(Z_i\) is not the output function and goes through several AND gates or OR gates but not any NOT gates then the function \(Z_a\) will still be in the form \(Z_a = g_i^* Z_1 + g_j^* Z_2 + g_k^* Z_3 + Z_4\) with \(Z_i Z_1 \leq Z_4\). Once \(Z_i\) enters a NOT gate, then the resultant function \(Z_i\) is in the form \(Z_i = \overline{g_i^* Z_a + g_j^* Z_a + Z_4}\).

(2) Suppose \(g_i^*\) and \(g_j^*\) first meet at an OR gate as indicated in Fig. 8(b), we have \(Z_i = g_i^* Z_1 + g_j^* Z_2 + Z_4\) which is in the form \(Z_i = g_i^* Z_1 + g_j^* Z_2 + Z_4\). If \(Z_i\) goes through several AND gates or OR gates but not any NOT gates, then the resultant function
$Z_u$ is still in the form $Z_u = g_1 Z_m + g_2 Z_a + Z_i$. Once $Z_u$ enters a NOT gate, then the resultant function is $Z_o = g_1 g_3 Z_i + g_2 Z_m Z_i + g_3 Z_a Z_i + Z_m Z_i$, which is in the form $Z_o = \bar{g}_1 g_3 Z_i + \bar{g}_2 Z_a + Z_i$, with $Z_o \leq Z_i$. If $Z_o$ goes through several AND gates or OR gates but not any NOT gates, the resultant function $Z_w$ remains in the form $Z_w = \bar{g}_1 g_3 Z_i + \bar{g}_2 Z_a + Z_i$ with $Z_w \leq Z_o$. Once $Z_w$ enters a NOT gate then the resultant function will be in the form $Z_r = g_1 g_3 Z_m + g_2 Z_m + Z_i$. (1) and (2) together show that if the output function of a fan-out free network is in the form $Z = g_1 g_3 Z_i + g_2 Z_a + g_2 Z_a + Z_i$, then the relation $Z_o \leq Z_i$ holds.

REFERENCES