EFFICIENT TEST GENERATION FOR CMOS CIRCUITS

Danu Radhakrishnan† and Congmin Lai†
†NETECH Corporation, 60 Bethpage Dr.,
Hicksville, New York 11801, U.S.A
‡Department of Electrical Engineering
University of Idaho, Moscow, ID 83843, U.S.A

Abstract

A new concept, path testing approach, is introduced in this paper. This approach leads to the simultaneous testing of a set of MOS transistors in a path of a CMOS circuit instead of individual ones. Using this concept, we generate a Karnaugh map procedure to identify a minimal robust test set for a direct CMOS complementary gate. The test set derived is implementation independent, and it can be used for detecting both single and multiple faults.

1 Introduction

Digital technology has gone through many dramatic changes in the last two decades. The advent of CMOS has provided a dominant technology for implementing VLSI circuits at an ever-increasing rate. In solving problems concerned with their testing.

The stuck-open fault model, which was first proposed by Wadack, is a very important fault model for testing CMOS circuits. Unlike a stuck-at fault, in the presence of a stuck-open fault, a CMOS combinational circuit behaves like a sequential circuit [1]. This not only increases the complexity of testing but also invalidates some of the test patterns [2]. To test a stuck-open fault, earlier researchers considered the MOS transistors one at a time [3,4]. This makes the test set very large and hard to identify.

In this paper we introduce a new concept, called path testing, which leads to the simultaneous testing of a set of MOS transistors in a path instead of individual ones.

2 CMOS Gate Networks

The design of CMOS combinational circuits is presently done by an interconnection of nMOS and pMOS transistors rather than by a number of primitive NAND/NOR gates. These circuits are called CMOS complex gates. In this paper a CMOS gate always means a CMOS complex gate.

A block diagram of a CMOS gate is given in Figure 1. It consists of a network of pMOS transistors, called the P-net, which is connected between the output node and \( V_{DD} \), and a network of nMOS transistors called the N-net which is connected between the output node and \( V_{SS} \). The capacitor shown represents both the load and stray capacitances, which introduces sequential circuit behavior in an otherwise combinational circuit, under fault conditions [1].

A variety of CMOS gate networks are available for any switching function. They include CMOS complementary logic, Pseudo nMOS logic, Dynamic CMOS logic, CMOS Domino logic, and Cascade Voltage Switch logic (CVSL) [5,6,7]. Only CMOS complementary gates are considered further in this paper.

The P-net and N-net in a CMOS complementary gate can be obtained from the K-map; the N-net from the 0's and the P-net from the 1's. This is done by first finding the partial pass functions \( f(0) \) and \( f(1) \) directly from the K-map in an optimum manner. For a more detailed discussion on pass networks, the reader is referred to [8]. \( f(0) \) and \( f(1) \) directly implement the N-net and P-net respectively. The P-net expression \( f_P \) is obtained from \( f(1) \) by dropping the pass variable \( I \) and complementing all literals. In a similar manner, the N-net expression \( f_N \) is obtained from \( f(0) \) by dropping the pass variable \( 0 \). \( f_P \) and \( f_N \) can be factored to minimize the number of transistors. Other methods of implementation make use of the dual relationship between the P-net and the N-net. Thus, CMOS complementary gates can be designed in four different ways. They are:

1. Design the N-net and P-net from \( f(0) \) and \( f(1) \) respectively.
   This is referred to as direct CMOS implementation in this paper.

2. Design the N-net from \( f(0) \) and its dual for P-net. This is referred to as dual p CMOS implementation.

3. Design the P-net from \( f(1) \) and its dual for N-net. This is referred to as dual n CMOS implementation.

4. Design the N-net and P-net from \( f(1) \) and \( f(0) \) respectively.
   This is referred to as dual CMOS implementation.

Normally, a minimal transistor implementation falls within the category of one of these dual n or dual p CMOS complementary gate.

3 Stuck-open Fault Test Generation

To test a stuck-open fault, a pair of test vectors are needed, the initialization vector \( T_1 \) initializes the circuit output, and the test vector \( T_2 \) tests the fault. This is called a two pattern test [1]. Some test patterns are invalidated due to circuit delays and time skews in circuit input changes. A test pattern which is never invalidated is called a robust test pair [2]. Earlier researchers always considered the testing of the MOS transistors one at a time [3,4]. The method presented in this paper, on the other hand, considers the simultaneous testing of a set of transistors in each path.

The following definitions will help the reader to understand the rest of the paper.

Definition 1. A path is a series connection of pMOS (nMOS) transistors connected between the output node and \( V_{DD} (V_{SS}) \). A path formed by pMOS transistors is called a p-path and a path formed by nMOS transistors is called an n-path.

---

This work was supported in part by the NASA Space Engineering Research Center under grant NAGW-1406

CH2819-1/90/0000-0588$00.80 ©1991 IEEE
Definition 2 A MOS transistor stuck-open fault is a failure that will leave one or more transistors permanently in a nonconducting state. Single stuck-open faults are those faults which force one MOS transistor to a permanently off state.

Definition 3 [4] Let \( P_i \) be a prime implicant in an irredundant sum of products expression, \( f = P_1 + P_2 + \ldots + P_i + \ldots + P_s \), of prime implicants of a function \( f \). Then a 1 vertex \( X \) of \( f \) is called a 1 ward of \( P_i \) if and only if \( X \) is covered by \( P_i \) and is not covered by any other prime implicant \( P_j \) of \( f \), \( P_j \neq P_i \), \( 1 \leq j \leq s \).

Definition 4 A 1 ward of a non-essential prime implicant is called a prime 1 cell.

In a direct CMOS complementary gate realizing a function \( f \), each and every p-path (n-path) corresponds to a prime implicant with complemented literals of \( f \) (prime implicant of \( f \)). A K-map of the function \( f \) can now be used to identify the robust test pairs for these paths. The necessary and sufficient conditions for a path in a direct CMOS complementary gate to be robustly testable are given in the following lemma.

Lemma 1 A p-path (n-path) in a direct CMOS complementary gate for \( f \) is robustly testable if and only if there exists at least one 1 ward of its corresponding prime implicant of \( f \) with at least one adjacent 0 cell.

Proof: To prove the sufficiency condition, assume that there exists a 0 cell adjacent to a 1 ward of the prime implicant corresponding to the p-path. Now initialize the gate to this 0 cell of \( f \). Next apply the 1 ward as the test vector. These two input vectors differ in only one variable. In a direct CMOS complementary gate, none of the p-paths have two transistors with one controlled by a variable and the other by its complement. So the p-path is robustly testable.

To prove the necessary condition, assume that there is no 0 cell adjacent to a 1 ward of the prime implicant of \( f \) corresponding to the p-path to be tested. Now, if a minterm of the above prime implicant is chosen as the test vector, which is not a 1 ward, then some other p-paths corresponding to the prime implicants of \( f \) which also cover that minterm will be enabled when this test input is applied. This will invalidate the test. On the other hand, if a 1 ward is applied as the test vector, then some other p-paths could be enabled temporarily during the transition between the initializing vector and the test vector due to arbitrary delays in the input changes, thus invalidating the test. Hence no robust test pair exists for this p-path.

In a similar manner the testability of an n-path can be proven.

Q E D

The above lemma implies that a direct CMOS complementary gate can be tested using one test pair per path. In the different p-paths (n-paths), there is at least one pMOS (nMOS) transistor different from each other. To test these transistors the test pairs must be different from each other. So the number of test pairs cannot be less than the number of paths. This implies that the test set generated by selecting one test pair for each and every path in a direct CMOS complementary gate is minimal. Furthermore, it forms a multiple stuck-open fault test set for the gate.

Unfortunately, for a few functions, none of the 1 wards of some prime implicants of all irredundant sum of products expressions of \( f \) has an adjacent 0 cell. Hence such functions are not robustly testable. For four variables only 32 functions do not have a robust test out of a total of \( 2^{16} \) [3]. The following lemmas and theorems help to identify a minimal robust test set for a direct CMOS complementary gate for a testable function.

Lemma 2 For a non-trivial function, all the essential minterms must have at least one adjacent maxterm.

For an irredundant combinational function implemented as a direct CMOS complementary gate, the stuck-open fault test set will be independent of its implementation if the test pairs belonging to the test set are invariant. For essential prime implicants, the test vectors can be selected the same for all implementations. Hence the test set will be implemented independent only if the test vectors for the non-essential prime implicants are invariant. To robustly test a path corresponding to a non-essential prime implicant, at least one prime 1 cell adjacent to a 0 cell must exist for each one of these prime implicants. It has been found that such a prime 1 cell exists for each non-essential prime implicant in an irredundant sum of products expression of an n variable function (n = 2, 3, 4, and 5) if the number of prime implicants covering that cell is equal to or less than \( n - i \), where \( 2^i \) is the number of cells covered by the largest prime implicant. Furthermore, it is found that these prime 1 cells are invariant. However, a search for an example to disprove the above claim has been unsuccessful. Hence the following conjecture.

Conjecture 1 If a prime 1 cell \( V_j \) is covered by \( n - i \) prime implicants or less, in an irredundant sum of products expression of an n variable function, then it is a prime 1 cell of all the other irredundant sum of products expressions of this function, and is always adjacent to a 0 cell, where \( 2^i \) is the number of cells covered by the largest prime implicant covering \( V_j \).

Theorem 1 For each and every non-essential prime implicant of an irredundant sum of products expression of \( f \) and \( \overline{f} \) if there exists at least one prime 1 cell satisfying Conjecture 1 such that no two of them taken together belong to any of the prime implicants of \( f \) and \( \overline{f} \) respectively, then all the stuck-open faults in the direct CMOS implementation of \( f \) are robustly testable, and the cardinality of the test set equals the total number of different paths in the gate.

Proof: An irredundant sum of products expression for \( f \) and \( \overline{f} \), in general, consists of a number of essential and non-essential prime implicants. From Lemma 2, all essential minterms have at least one adjacent 0 cell. Hence the p-paths (n-paths) corresponding to all essential prime implicants of \( f \) and \( \overline{f} \) are robustly testable using the essential minterms of \( f \) as test vectors and their adjacent 0 cells as the initialization inputs.

The total number of non-essential prime implicants in any irredundant sum of products expression is a constant. For the stuck-
open fault testing of the path corresponding to each non-essential prime implicant, one test vector is included in the test vector set. Any prime 1 cell, adjacent to a 0 cell, covered by that prime implicant can be selected as the test vector. In this case, the prime 1 cells form the test vectors and their adjacent 0 cells form the initialization inputs. For a direct CMOS complementary gate if we can find at least one prime 1 cell, adjacent to a 0 cell, for each and every non-essential prime implicant then there exists a complete set of stuck-open fault tests for that gate. If, in addition, the prime 1 cells selected satisfy Conjecture 1, then these prime 1 cells always stay as prime 1 cells.

Any implementation must cover all these prime 1 cells. Since no two of them are covered by any of the prime implicants, they always belong to different prime implicants in any implementation. Hence all direct CMOS complementary gate implementations are robustly testable.

The total number of different paths (both p and n) in the circuit equals the sum of the prime implicants in the irredundant sum of products expressions of f and \( \bar{f} \) put together.

The test vectors for the essential prime implicants do not change. For the non-essential prime implicants, the prime 1 cells are invariant with the implementation. Hence their test vectors are also invariant. This makes the total test set invariant. Since the test set consists of one test vector pair for each of the prime implicants of f and \( \bar{f} \), the cardinality of the test set equals the total number of paths in the gate.

Theorem 1 above gives only the sufficient conditions for finding a complete set of stuck-open fault tests which is implementation independent. There may be cases where the test set is implementation dependent. So far we are unable to find an example where the test set varies with implementation. Also, we are unable to give a general proof to show that the test set is implementation independent.

3.1 Test generation using a Karnaugh map

For functions up to 6 variables, Procedure 1 enumerates the steps involved in finding a complete set of robust test vectors which is implementation independent. This procedure is based on Theorem 1.

Procedure 1

1. Form the K-map for the function f.
2. Identify the set of all prime implicants of f.
3. Select one essential minterm for each essential prime implicant of f.
4. Select one prime 1 cell which is covered by at most \( n - i \) prime implicants for each of the non-essential prime implicants in an irredundant sum of products expression of f, such that no two of them are in the same prime implicant, which belongs to the prime implicant set of f identified in Step 2.

5. The set of all \( i \) cells of f from Steps 3 and 4 form the set of test vectors in the two pattern test for the P-net.
6. Repeat Steps 1 to 4 by changing f to \( \bar{f} \) to find the test vector set for the N-net.

The following example illustrates the generation of a stuck-open fault test set using the above procedure.

Example 1 Consider the function

\[
f(A, B, C, D) = \sum_j(0, 2, 5, 6, 7, 8, 10, 12, 13, 14, 15)
\]

The K-map for the function is shown in Figure 2. The set of all possible prime implicants of f is \{B'D', AB, BD, BC, CD', AD'\}. One choice of prime implicants for a minimal expression for f is encircled in the K-map. There are 2 essential prime implicants. They are BD and B'D'. The essential 1 cells are marked with a "x". The non-essential prime implicants are AB and BC. The prime 1 cells in these prime implicants are marked with a "+".

These prime 1 cells together do not belong to any of the prime implicants in the prime implicant set given above. Each one of these prime 1 cells is covered in prime implicants of size 4 (group of 4 cells) and has only two possible prime implicant coverings. A minimal expression for the P-net of a direct CMOS complementary gate is given by:

\[
f_p = B'D' + BD + A'B' + B'C'
\]

(1)

A minimal robust test set for the P-net includes one test vector for each prime implicant. The prime implicants and their corresponding test vectors are tabulated in Table 1. For the prime implicant BD the test vector in the test vector pair is unique. It is the essential 1 cell 001. For this 1 cell, there are 2 choices for the initialization vectors. They are 0001 and 0100. Similarly, the test vector pairs for the other essential prime implicant B'D' are (0001, 0000) or (0100, 0000). For the non-essential prime implicant BD, the prime 1 cell 1100 has an adjacent 0 cell 0100 and hence the test vector pair is (0100, 11000). Similarly, the test vector pair for the second non-essential prime implicant BC is (0100, 0110).

The last two terms A'B' and B'C' in Equation 1 have other choices. They are A'D and C'D. Thus the four choices for the implementation of the P-net are:

\[
\begin{align*}
f_p &= B'D' + BD + A'B' + B'C' = B'(D' + A' + C') + BD \\
f_p &= B'D' + BD + C'D + A'D = D(B + C' + A') + B'D' \\
f_p &= B'D' + BD + C'D + A'B' = B'(D' + A') + D(B + C'), \\
\end{align*}
\]

and

\[
f_p = B'D' + BD + B'C' + A'D = B'(D' + C') + D(B + A')
\]

This gives 4 possible P-net implementations for a direct CMOS complementary gate. The pMOS stuck-open fault test set shown in Table 1 is valid for all these four implementations.

The N-net expression is given by:

\[
f_N = B'D + A'BC'D'
\]

(2)

The tests for the N-net are found in a manner similar to the ones for the P-net. These test vectors are tabulated in Table 2. The
*N-net in the direct CMOS implementation is unique. Hence a total of 4 possible direct CMOS implementations exist for f.*

The generation of stuck-open fault tests using a K-map is possible only for functions up to six variables. For functions of more than six variables an algorithmic approach is necessary. This is done by a procedure similar to the Quine-McCluskey tabular approach used for the minimization of switching functions.

4 Conclusion

A procedure to identify a robust test set for a direct CMOS complementary gate using a K-map is given in this paper. This procedure first provides a means to identify whether a direct CMOS complementary gate of a function is completely testable for all its stuck-open faults. Then it provides a means to identify an implementation independent stuck-open fault test set if one exists. Furthermore, the stuck-open fault test set generated using the procedure above gives a minimal test set for a direct CMOS complementary gate. This procedure can also be used to identify a stuck-open fault test set for the P-net of a dual n CMOS complementary gate and for the N-net of a dual p CMOS complementary gate.

References


