VLSI DESIGN
1996, Vol. 5, No. 1, pp. 59–75
Reprints available directly from the publisher
Photocopying permitted by license only

A New Theory for Testability-Preserving Optimization of Combinational Circuits

JIABI ZHU[a], MOSTAFA ABD-EL-BARR[b], and CARL McCROSKY[b,*]

[a]*Silicon Valley Research, 300 Ferguson Drive, Suite 300, Mountain View, CA 94043, U.S.A.;* [b]*Dept. of Computational Science, Univ. of Saskatchewan, Saskatoon, Sask. CANADA S7N 0W0*

(Received 29 October 1994; In final form 30 January 1995)

Testability should be considered as early as possible in VLSI synthesis and optimization. In most CAD tools, testability preservation is achieved as a by-product of finding a less redundant implementation. Unfortunately, this approach is not supported by theory. It is essential to have an optimization scheme that systematically preserves testability. A complete theory covering testability-preserving optimization of combinational circuits is yet to be developed.

In this paper, a new testability-preserving theory is established. This theory covers most optimization cases in the synthesis of irredundant circuits. Optimization of redundant circuits is also considered. Algorithms based on this theory are proposed. Experiments to verify the theory and algorithms are reported. The developed theory and algorithms are more general than previous results.

Keywords: Circuit optimization, gate level circuits, factorization, substitution, testability-preserving optimization

1. INTRODUCTION

With increasing complexity of VLSI (Very Large Scale Integrated) circuits, the design of chips is increasingly dependent on CAD tools for automatic synthesis, layout, and testing. These tools reduce the time required to design needed circuits.

Synthesis tools should produce minimal or nearly minimal logic (or transistor) circuits in order to reduce manufacturing costs [1]. At the same time, it is important that the resultant circuits be testable in order to reliably and efficiently detect manufacturing faults [2, 3]. Unfortunately, as will be shown in Section 3.2 of this paper, optimization of a circuit may cause faults which were testable in the unoptimized circuit to be no longer testable in the optimized circuit.

In order to support the twin goals of optimization and testability, it is useful to develop optimization techniques which preserve testability.

Rajski and Vasudevamurthy [4] studied testability preservation of some special categories of irredundant circuits using a factorization algorithm. However, their concepts and theorems did not cover re-

*Corresponding author. Tel.: (306) 966-4896. Fax: (306) 966-4884. E-mail: carl.mccrosky@usask.ca.

dundant circuits. A more comprehensive theory for testability-preserving optimization of combinational circuits is needed, and its presentation should include experimental results.

In this paper, a new testability-preserving theory is established. This theory covers optimization of redundant as well as irredundant circuits. An optimizing algorithm for redundant and irredundant circuits is proposed. Experimental results are shown as a verification of the validity of the theory and the algorithm proposed.

The paper is organized as follows. In Section 2, existing circuit optimization techniques are reviewed. In Section 3, recent research in testability-preserving optimization is introduced. In Section 4, a new theory for testability-preserving optimization is proposed and proven. An algorithm based on this theory is also introduced. Section 5 shows the experimental results obtained from applying the theory and the algorithm. Section 6 gives a conclusion and discusses open research topics in this area.

2. CIRCUIT OPTIMIZATION METHODS

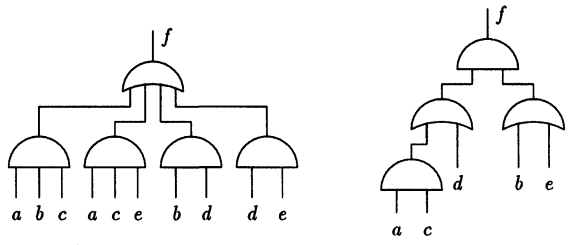
Usually, factorization and substitution approaches are used in gate level circuit synthesis. These techniques are presented and their features are discussed in the following subsections.

2.1 Factorization Methods

The factorization approach [5, 6, 7, 8, 9, 10] is a circuit optimization method which uses Boolean algebra operations/transformations to reduce the number of literals in Boolean expressions. Factorization results in a (near) minimum factored expression for a given function. Consider, for example, the function

$$f = abc + ace + bd + de$$

The corresponding gate-level realization of the function f is shown in Figure 1(a). The Boolean expression of the function can be factorized into



(a) Original circuit (b) Factorized circuit

FIGURE 1 Factorization method

$$f = (ac + d)(b + e)$$

The realization of the factorized form of the function is shown in Figure 1(b).

The main disadvantage of factorization is that extraction of factors is constrained to be among variables at the same level. If paths from the terms that can be extracted to the output of the circuit are drawn, it can be found that these paths must merge at the gate in the next logic level or the second logic level thereafter (in Figure 1(a), the paths from two input terms "ac" to the output f merge at the OR gate in the second logic level). In general, circuits cannot be further optimized due to this limitation. This limitation is further illustrated in the following example.

Example 1 Consider the following Boolean expression.

$$a(bcd + e) + g(bch + i)$$

The multilevel circuit realization of this expression is shown in Figure 2. Although the two terms "bc" are in the same level in the circuit, they cannot be ex-

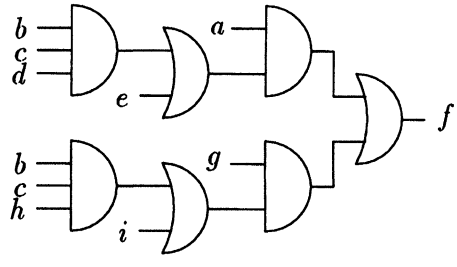


FIGURE 2 Limitation of factorization

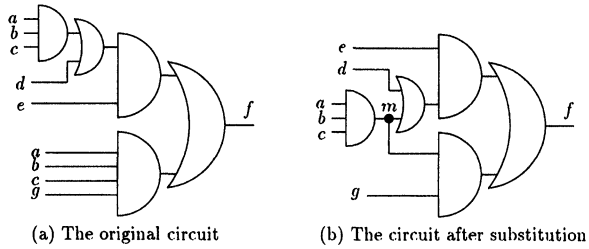
tracted for optimization because the paths from these two terms to f merge at the next fourth level thereafter.

2.2 Substitution Methods

Substitution is another method for optimizing circuits. The basic idea is to find common parts (containing two or more literals) which occur two or more times in an expression, and substitute these parts with a single temporal signal line (variable). After the substitution, a fan-out from the single sub-circuit which generates a Boolean function for the common parts in the original circuit is created. By replacing the duplicated parts with the shared sub-circuit, gates in the circuit can be saved. A simple example is shown in Figure 3. Here, the substitution $m = abc$ is made.

In the substitution methods, extractions are allowed among the signal lines (variables) in different levels, and extracted signal lines can merge in any manner. In most cases, the substituted circuits have fewer connections as well as fewer and/or smaller gates. Substitution reduces the duplication in both gates and connections. Therefore, in addition to using fewer and smaller gates, the substituted circuit uses less area for routing wires.

The main disadvantages of substitution methods is that the computing cost is higher than that of simple factorization methods. In the substitution method, there are more possible extractions available and more possible plans for substitution have to be evaluated.



(a) The original circuit (b) The circuit after substitution

FIGURE 3 Substitution method

3. TESTABILITY-PRESERVING OPTIMIZATION

There are numerous possible definitions for testability. For this paper we require a specific meaning for the this term. Our definition of testability depends upon a specific fault model. We assume that all faults can be modeled as stuck-at-zero (s-a-0) or stuck-at-one (s-a-1) faults; further we assume that only one such fault exists in the circuit-under-test. Now we define *testability* to be the number of possible faults which cannot be detected at the primary outputs of a circuit with any vector applied to the primary inputs of the circuit. Our objective is to reduce this number, a testability measure of zero is ideal.

It is significant that circuit testability be preserved, or even improved during synthesis. In this section, the basic concepts, background, and recent research in the area of testability-preserving optimization are introduced.

3.1 Redundant Circuits and Undetectable Faults

The single stuck-at-fault model [11] is suitable for gate level circuit model of combinational circuits. This model is used in this paper.

If there exists no test vector (i.e., a given assignment to the set of inputs) which can detect some given fault ϕ in a circuit, then the circuit is said to be *redundant* with respect to ϕ. Consider the circuit shown in Figure 4. This circuit is redundant with respect to the s-a-0 fault on line 3, denoted as ϕ_1, since the occurrence of this fault does not change the function realized at f. Thus it is tempting to just ignore such faults. However, the presence of undetectable faults can cause some detectable faults to become un-

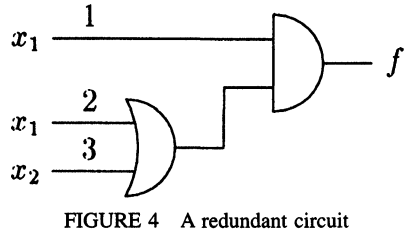
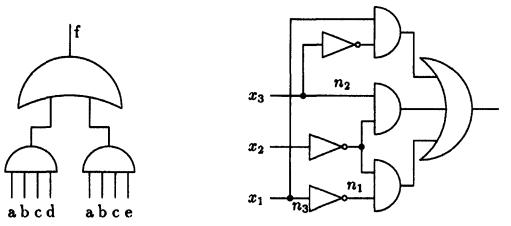


FIGURE 4 A redundant circuit

detectable. For example, undetectable fault ϕ_1 can cause s-a-1 fault on line 1 to be undetectable [12]. It should be noted that line 1 was detectable in the absence of fault ϕ_1.

Another problem that arises in dealing with redundant circuits is that part of the computational effort is wasted in trying to find a test vector for an undetectable fault.

Therefore it is advisable to remove any redundant part(s) of a circuit in the design phase itself. Unfortunately, for larger circuits identifying the redundant lines and gates can be a very time-consuming process. Therefore extra care needs to be taken while designing a circuit to avoid redundancy [12].



(a) Redundant in terms of optimization but irredundant in terms of testing (b) Irredundant in terms of optimization but redundant in terms of testing

FIGURE 5 Two different redundancy definitions

3.2 The Relationship between Optimization and Testability

Generally speaking, the less redundant a circuit is, the better its testability. In other words, redundancy can be a reason that a given circuit is less testable. A circuit is optimized by reducing its redundancy. In most existing automatic synthesis systems, testability is considered only as a side effect of a less redundant implementation [10].

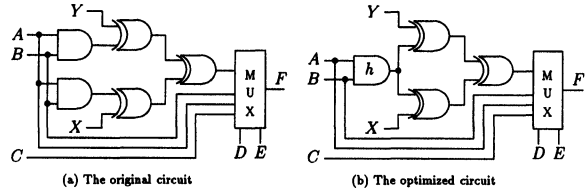
However, this approach is supported by neither a theory nor experimental results. The redundancy concept in the context of *circuit optimization* and the redundancy definition in the context of *circuit testing* should not be confused. In circuit optimization, the components (gates, transistors, and lines) are said to be redundant if they can be removed, for example, through optimization, without changing the logic function realized by the circuit. Some circuits are *redundant* in terms of circuit optimization but are *irredundant* in terms of testing, see, for example, the circuit shown in Figure 5(a). Some other circuits, see, for example, the circuit shown in Figure 5(b), are *irredundant* in the context of circuit optimization but are *redundant* in the context of testing.

In practice, the traditional approach, which tries to reduce the redundancy in terms of testing by reducing the redundancy in terms of optimization, cannot guarantee testability preservation. In some cases, optimization may make some detectable faults in the original circuit undetectable. An example can be found in [13], as shown in Figure 6. The original circuit in Figure 6(a) is fully testable, but gate h in the optimized circuit in Figure 6(b) is not testable.

3.3 The Testability-Preserving Factorization Algorithm by Rajski and Vasudevamurthy

It has been shown in [4] that if a network (two-level or multilevel) is completely testable, then the network obtained by any of the Boolean algebraic transformations along with De Morgan's law is also completely testable. Rajski and Vasudevamurthy gave the following definition for testability preservation.

DEFINITION 1 Let N_1 be a network and F be a transformation which transforms N_1 to a network N_2. Let T be a complete test set for single stuck-at faults for N_1. We say that transformation F **preserves testability**, if T is also a complete test set for single stuck-at faults in network N_2.



(a) The original circuit (b) The optimized circuit

FIGURE 6 Optimization may make some faults undetectable

The results given by Rajski and Vasudevamurthy in [4] could be summarized in the following theorem where part 1 corresponds to Theorems 3–5 in [4], and part 2 corresponds to Theorem 6 in [4].

THEOREM 1

(1) In factorization, the transformations along with De Morgan's law preserve testability, if the original circuit is irredundant.

(2) Resubstitution of common sub-expressions in a multi-output irredundant circuit, as shown in Figure 7, preserves testability if no two sub-expressions control the same output, i.e., X cannot reach P_i via y_j, and cannot reach P_j via y_i.

Rajski and Vasudevamurthy opened a new research area. Their theory provides a systematic solution to the testability preservation problem in combinational circuit optimization. However, we have the following observations to make:

1. **Applicability**: Theorem 1 is only proved to be applicable to irredundant circuits. In fact, nearly half of the circuits used in the experiments in [4] were redundant, and therefore were not covered by the above theorem.
2. **Comprehensiveness**: Theorem 1 includes four cases used in factorization of irredundant circuits. Other operations, for example, the sub-expression substitution shown in Figure 3, was not considered. As will be shown in Theorem 3 of this paper, there is room for improvement over the above theorem for irredundant circuits.
3. **Reported experimental results**: Some experimental results which demonstrated the efficiency in terms of optimization of the factorization algorithm were reported in [4]. However, no data regarding the testability preservation of these results was given.

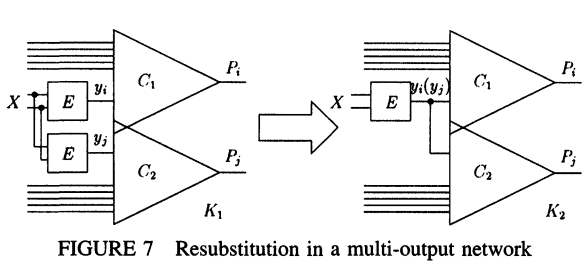


FIGURE 7 Resubstitution in a multi-output network

4. E/O APPROACH TO TESTABILITY-PRESERVING OPTIMIZATION

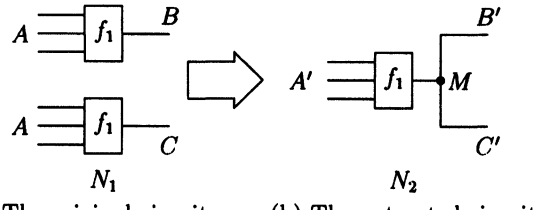
In this section, a more general new theory for testability-preserving optimization—E/O approach—is proposed. In the following discussion, "E" means even, and "O" means odd.

4.1 Definitions and Concepts

The denotation D from the D-algorithm [14] is used in this work. The symbol D is used to denote the situation in which a line has a value 1 in the absence of the fault and 0 in its presence. The complementary situation is denoted by $\bar{D}$. For example, a s-a-0 fault on line x_1 is represented as fault $x_1 = D$; a s-a-1 fault on line x_1 is represented as fault $x_1 = \bar{D}$.

We define the concept of testability-preserving optimization such that it is applicable to both irredundant and redundant circuits.

DEFINITION 2 Let N_1 be a network and F a transformation (or extraction) which transforms N_1 to a network N_2, as shown in Figure 8. Suppose that transformation F removes the copy of f_1 which produces C, and creates fan-out point M connected with B' and C'. It is said that transformation (or extraction) F **preserves testability,** if the following conditions are true:



(a) The original circuit (b) The extracted circuit

FIGURE 8 Testability preserving transformation

1. If fault B = D (or C = D) is detectable in N_1, then single faults B' = D (or C' = D) and M = D will also be detectable in N_2, and
2. If fault B = $\bar{D}$ (or C = $\bar{D}$) is detectable in N_1, then single faults B' = $\bar{D}$ (or C' = $\bar{D}$) and M = $\bar{D}$ will also be detectable in N_2.

In order to derive the theory of testability preservation for optimization of redundant circuits as well as irredundant circuits, Lemma 7 in [4], which is only applicable to irredundant circuits, should be generalized to be applicable to redundant circuits as well. Suppose that a network N_1 is transformed to N_2 in a step of an optimization sequence, where N_1 can be redundant or irredundant. Only a part of the network N_1, namely K_1, is transformed to K_2 in N_2, and K_1 and K_2 perform the same function. We have the following lemma.

LEMMA 1 *A test set T detects a fault set S_f of single stuck-at faults in N_1, outside of K_1, ($N_1 \backslash K_1$), if and only if T detects all corresponding stuck-at faults of S_f in N_2, outside of K_2, ($N_2 \backslash K_2$), as shown in Figure 9.*

The proof is similar to the proof of Lemma 7 in [4].

With this lemma, in proving the testability preservation of a transformation, it is sufficient to demonstrate that any line in K_2 is testable if the corresponding line in K_1 is testable.

4.2 E/O Theory

To simplify the discussion, we will assume that a given circuit is decomposed in terms of AND, OR and NOT gates unless otherwise indicated. For example, a NAND gate is decomposed to an AND gate followed by a NOT gate.

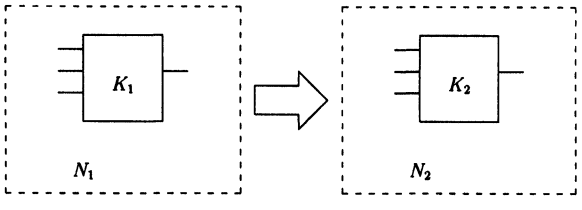


FIGURE 9 Transformation of redundant/irredundant networks

The effect of a fault in a circuit may propagate to the primary output of the circuit through one or more paths.

DEFINITION 3 The **E/O value of a path** is defined as even (or odd) if the number of inversions (i.e., inverter gates) on the path is even (or odd).

DEFINITION 4 Suppose that there exist more than one path from node n_1 to node n_2 in a circuit. If the E/O values of these paths are not the same, i.e., some are even, while the others are odd, then it is said that there is a **conflict**; otherwise, we define the **E/O value of the paths from n_1 to n_2** as the common E/O value of these paths. If the E/O value from n_1 to n_2 is even, this relationship is denoted as:

$$n_1 \xrightarrow{E} n_2;$$

If the E/O value from n_1 to n_2 is odd, this relationship is denoted as:

$$n_1 \xrightarrow{O} n_2.$$

According to Definition 4, if the *E/O* value of the primary output is fixed, for example, as *E*, then the *E/O* value of each node in the circuit can also be calculated unless the node is involved in conflict. The process is from the primary output towards the primary inputs. During this backward scan, an *E/O* value is maintained until an inverting gate (NOT, NOR, or NAND) is encountered. The *E/O* value is updated from *E* to *O* or from *O* to *E* at the input line(s) of the inverting gate. This backward scan continues until the primary input is encountered.

Example 2 Consider the circuit shown in Figure 10. To mark the E/O value of each node in the circuit, we mark the primary output line 11 as E and proceed from the primary output towards the primary inputs. Input lines 9 and 10 of gate G_5 are marked as E because G_5 is a non-inverter gate. However, input lines 7 and 8 of G_4 are marked as O because G_4 is an inverting gate. The E/O value is updated again at G_2

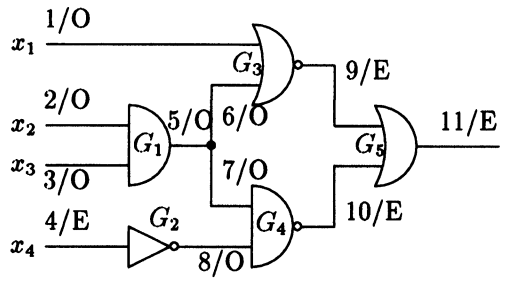


FIGURE 10 E/O values of the nodes in a circuit

from O at its output line 8 to E at its input line 4. In a similar way, the rest of the nodes can be marked as shown in Figure 10.

DEFINITION 5 The **arithmetic difference** of a function f with respect to an input x_i is defined as:

$$\frac{\partial f}{\partial x_i} = f(x_i = 1) - f(x_i = 0),$$

where "−" is the arithmetic subtraction operator.

The arithmetic difference can be 0, 1, or −1, i.e.,

$$\frac{\partial f}{\partial x_i} \in \{0, 1, -1\}.$$

The meaning of $\partial f / \partial x_i = 0$ is that f remains unchanged due to a change in x_i. $\partial f / \partial x_i = 1$ means that f changes in the same direction as the change in x_i, i.e., when x_i changes from 1 to 0 (or 0 to 1), f changes from 1 to 0 (or 0 to 1); $\partial f / \partial x_i = -1$ means that f changes in the opposite direction to a change in x_i, i.e., when x_i changes from 1 to 0 (or 0 to 1), f changes from 0 to 1 (or 1 to 0). In terms of testing, $\partial f / \partial x_i = 0$ means that the effect of a stuck-at fault ϕ on line x_i cannot propagate to f. If f is the only output of the circuit and $\partial f / \partial x_i = 0$ for all the inputs, then ϕ is considered undetectable. On the other hand, $\partial f / \partial x_i \neq 0$ means stuck-at fault ϕ on line x_i is detectable.

LEMMA 2 *Suppose that a circuit f consists of AND and OR gates, i.e., there are no NOT, NAND, NOR, or XOR gates in the circuit, then*

$$\frac{\partial f}{\partial p} \in \{0, 1\}.$$

where p is any point in the circuit.

Proof Because the functions OR and AND are monotonically increasing (or called as *unate* in [12, 15]), the circuit composed of OR and AND gates only is also monotonically increasing, i.e.,

$$\frac{\partial f}{\partial p} \geq 0.$$

LEMMA 3 *Suppose that there is a single-path from A to x_1 in the circuits shown in Figure 11(a)–(c). Then we have:*

$$\frac{\partial f_{OR}}{\partial A} = \frac{\partial x_1}{\partial A} \times \frac{\partial f_{OR}}{\partial x_1}. \tag{1}$$

$$\frac{\partial f_{AND}}{\partial A} = \frac{\partial x_1}{\partial A} \times \frac{\partial f_{AND}}{\partial x_1}. \tag{2}$$

$$\frac{\partial f_{NOT}}{\partial A} = -\frac{\partial x_1}{\partial A}. \tag{3}$$

where "×" is the multiplication operator.

Proof (1) If at least one of the inputs $\{x_2, x_3, \ldots, x_n\}$ of the OR gate is 1, then the effect of a fault on line A cannot propagate from A to f_{OR} through x_1. Hence, the left hand side of Equation 1 is $\partial f_{OR} / \partial A = 0$. On the other hand, the right hand side of Equation 1 is:

$$\frac{\partial x_1}{\partial A} \times \frac{\partial f_{OR}}{\partial x_1} = \frac{\partial x_1}{\partial A} \times 0 = 0.$$

When all inputs $\{x_2, x_3, \ldots, x_n\}$ of the OR gate are 0, then we have $f = x_1$. Therefore,

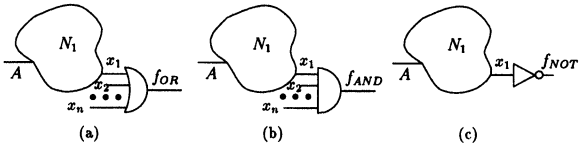


FIGURE 11 Illustration for the properties of gates

$$\frac{\partial f_{OR}}{\partial A} = \frac{\partial x_1}{\partial A}.$$

For the right hand side of Equation 1,

$$\frac{\partial x_1}{\partial A} \times \frac{\partial f_{OR}}{\partial x_1} = \frac{\partial x_1}{\partial A} \times 1 = \frac{\partial x_1}{\partial A}.$$

Hence, Equation 1 is correct.

(2) The proof for Equation 2 is similar to that for Equation 1.

(3) If x_1 is independent of A, then the effect of a fault on A cannot propagate to x_1 and hence to f_{NOT}, therefore both x_1 and f_{NOT} remain unchanged, i.e.,

$$\frac{\partial x_1}{\partial A} = 0$$

and

$$\frac{\partial f_{NOT}}{\partial A} = 0;$$

otherwise, the change at f_{NOT} is opposite to the change at x_1, i.e.,

$$\frac{\partial f_{NOT}}{\partial A} = -\frac{\partial x_1}{\partial A}.$$

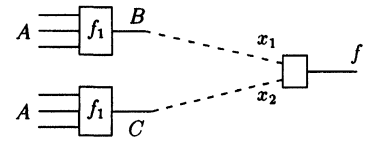
Hence, Equation 3 is correct. Q.E.D

In using substitution in single output redundant circuits, testability can be preserved if the condition in the following lemma is satisfied.

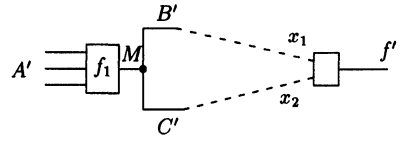
LEMMA 4 *In the circuit shown in Figure 12, suppose that the E/O values of nodes B and C are the same. Then the extraction from Figure 12(a) to Figure 12(b) preserves testability.*

Proof For the proof of Lemma 4, please refer to Appendix A. Q.E.D

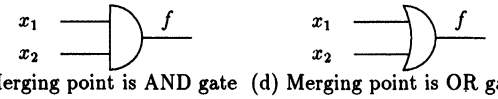
LEMMA 5 *Consider the substitution in Figure 13. Suppose that all paths from B and C to primary output f must pass line p, and that B and C have the same E/O values counted from p. Then the merge of B and C preserves testability.*



(a) The original circuit



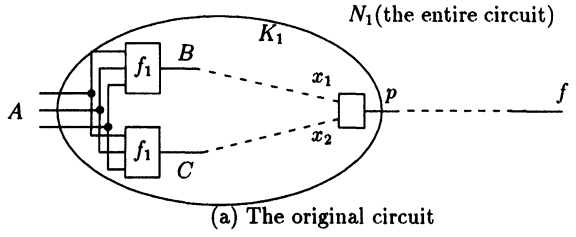
(b) The extracted circuit



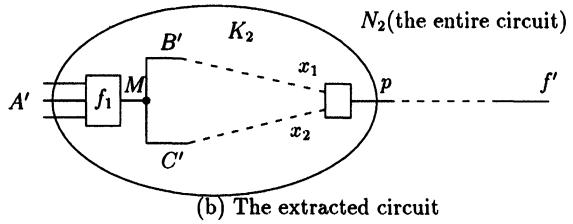
(c) Merging point is AND gate (d) Merging point is OR gate

FIGURE 12 Substitution procedure

Proof According to the assumption, the sub-circuit K_1 which is changed into K_2 during the substitution has only one output p. From Lemmas 1 and 4, we know that the substitution preserves testability. Q.E.D.



(a) The original circuit



(b) The extracted circuit

FIGURE 13 Substitution procedure

LEMMA 6 *The extraction of NOT gates from a gate (an AND or an OR gate), as shown in Figure 14, preserves testability no matter whether the original circuit is redundant or irredundant.*

Proof Each fault in K_2 corresponds to one fault in K_1 except that the type of the fault is inverted. For example, a stuck-at-1 fault on line l_1' in K_2 corresponds to a stuck-at-0 fault on the corresponding line l_1 in K_1. Therefore, if the stuck-at-0 fault on l_1 in K_1 is detectable, then the stuck-at-1 fault on line l_1' in K_2 is also detectable. Q.E.D.

LEMMA 7 *Double cube extraction using factorization, i.e.,*

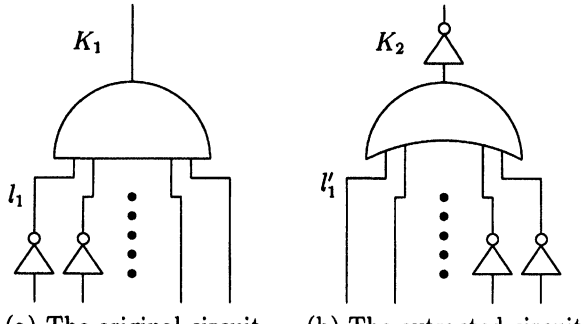
$$f = AS + AT + R$$
$$= A(S + T) + R$$

as shown in Figure 15, preserves testability for redundant circuits, where $A = a_1\, a_2 \ldots a_{n_a}$, $S = s_1\, s_2 \ldots, s_{n_s}$, *and* $T = t_1, t_2, \ldots, t_{n_t}$.

Proof We will prove that every line in the optimized circuit in Figure 15(b) is testable if the corresponding line in the original circuit in Figure 15(a) is testable.

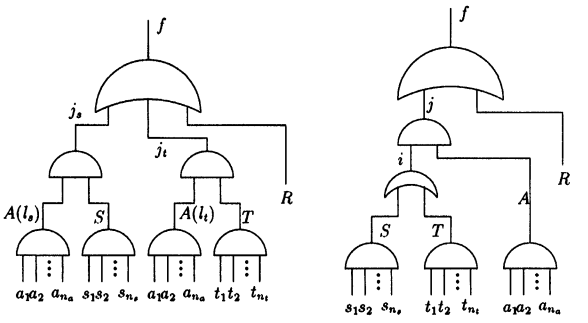
During the transformation, lines $j_s = AS$ and $j_t = AT$ in Figure 15(a) are removed, while new lines $i = S + T$ and $j = i\,A$ in Figure 15(b) are created.

(1) First, let us consider the testability of line j in Figure 15(b).



(a) The original circuit (b) The extracted circuit

FIGURE 14 Extraction of NOT gates



(a) The original circuit (b) The circuit after factorization

FIGURE 15 Double cube factorization

Suppose that s-a-0 fault on line j_s (or $j_s = D$) in the original circuit is detectable, i.e., there exists a test pattern T_1 such that when the input of the circuit is T_1, $j_t = 0$, $R = 0$, and $j_s = 1$ under fault-free condition. This is the condition that fault $j_s = D$ can propagate to f and be detected. Notice that $j_s = 1$ requires $S = 1$ and $A = 1$.

Now, let us apply T_1 to the optimized circuit in Figure 15(b). Because $S = 1$ and $A = 1$, then $j = 1$ in the optimized circuit under fault-free condition. Furthermore, because $R = 0$, $j_s = D$ can propagate to f. Therefore, $j = D$ in the optimized circuit is detectable.

Similarly, it can be shown that if $j_t = D$ in the original circuit is detectable, then $j = D$ in the optimized circuit is also detectable.

In the same way, it can be shown that if either $j_s = \bar{D}$ or $j_t = \bar{D}$ in the original circuit is detectable, then $j = \bar{D}$ in the optimized circuit is also detectable.

Therefore, testability of line j is preserved after optimization.

(2) Let us consider the testability of line i in Figure 15(b).

In a similar way as above, it can be shown that if either $j_s = D$ or $j_t = D$ is detectable in the original circuit, then $i = D$ in the optimized circuit is also detectable.

However, to detect fault $i = \bar{D}$ in the optimized circuit requires that $A = 1$, $S = 0$, $T = 0$ and $R = 0$. In fact, this condition is also necessary to detect fault $S = \bar{D}$ (or $s_k = \bar{D}$ where $1 \le k \le n_s$) and fault $T = \bar{D}$ (or $t_k = \bar{D}$ where $1 \le k \le n_t$) in the original circuit.

In other words, if either $S = \bar{D}$ (or $s_k = \bar{D}$) or $T = \bar{D}$ (or $t_k = \bar{D}$) in the original circuit is detectable, then $i = \bar{D}$ in the optimized circuit is also detectable.

Therefore, testability of line i is preserved after optimization.

(3) Let us consider the testability of line S in Figure 15(b).

In a similar way as (1), it can be shown that S in Figure 15(b) is testable if S in Figure 15(a) is testable.

(4) Let us consider the testability of line A in Figure 15(b).

In a similar way as (1), it can be shown that if either line l_s or l_t in Figure 15(a) is testable, then A in Figure 15(b) is also testable.

(5) Let us consider lines a_u, s_v, and t_w in Figure 15(b) where $u = 1, 2, \ldots, n_a$, $v = 1, 2, \ldots, n_s$, and $w = 1, 2, \ldots, n_t$.

Suppose that line s_v in Figure 15(a) is testable with test pattern T_2, i.e., the fault on s_v can propagate to f via S. In the same way, the fault on s_v in Figure 15(b) should also propagate to f via S because S is testable with T_2. Therefore, s_v in Figure 15(b) is testable.

In a similar way, it can be shown that line a_u in Figure 15(b) is testable if any a_u in Figure 15(a) is testable, and line t_w in Figure 15(b) is testable if t_w in Figure 15(a) is testable.

(6) Line R is unchanged in the transformation.

From (1) to (6), it can be concluded that double cube factorization preserves testability for redundant circuits. Q.E.D.

From Lemmas 1, 4, 6, and 7 we have Theorem 2. Theorem 6 in [4] can also be generalized for redundant circuits as shown in the following theorem.

THEOREM 2 *The following redundant circuits optimizations preserve testability:*

1. *The substitution of nodes B and C which represent the same Boolean function, as shown in Figure 13. Suppose that p is a line that all paths from B and C to all primary outputs must pass. Then the substitution preserves testability if one of the following conditions is*
   - *(a) B and C have the same E/O values counted from p (Lemma 4).*
   - *(b) B and C do not control the same primary output.*
2. *The extraction of NOT gates as shown in Figure 14 (Lemma 6).*
3. *The Double cube factorization as shown in Figure 15 (Lemma 7).*

If the original circuit is irredundant, i.e., fully testable, then substitutions of nodes with opposite *E/O* values also preserve testability, as stated in Lemma 9 below. Before showing Lemma 9, we need to introduce Lemma 8.

LEMMA 8 *Consider the substitution in Figure 12. Suppose that all paths from B and C to f meet only at f, and* $B \xrightarrow{E} f$, *and* $C \xrightarrow{O} f$. *The substitution preserves testability, if at least one of the following conditions is satisfied:*

1. *Gate f is an AND gate, and single faults $B = D$ and $C = \bar{D}$ in the original circuit are detectable.*
2. *Gate f is an OR gate, and single faults $B = \bar{D}$ and $C = D$ in the original circuit are detectable.*

Proof Please refer to Appendix B. Q.E.D.

LEMMA 9 *Consider the substitution nodes B and C in an irredundant circuit. Suppose that p is a line that all paths from B and C to the primary output f must pass. Then the substitution preserves testability if there is no conflict in the E/O values on paths* $B \rightarrow p$ *and* $C \rightarrow p$, *i.e., one of the following conditions is satisfied:*

1. *B and C have the same E/O values counted from p.*
2. *B and C have the opposite E/O values counted from p.*

Proof Case 1 is the direct result from Lemmas 4 and 1. Case 2 is the direct result from Lemmas 8 and 1. Q.E.D.

Lemma 9 can be generalized for multi-output circuits, as stated in Part 1 of Theorem 3.

THEOREM 3 *The following optimizations of irredundant circuits (single-output or multi-output) preserve testability:*

1. *The substitution of nodes B and C which represent the same Boolean function, as shown in Figure 13. Suppose that p is a line that all paths from B and C to all primary outputs must pass. Then the substitution preserves testability if one of the following conditions is satisfied:*
   (a) *All paths from B to p have the same E/O value; and all paths from C to p have the same E/O value (Lemma 4).*
   (b) *B and C do not control the same primary output.*
2. *The extraction of NOT gates shown in Figure 14 (Lemma 6).*
3. *The Double cube factorization shown in Figure 15 (Lemma 7).*

It should be noted that all the cases in the theorem due to Rajski and Vasudevamurthy (Theorem 1 in Section 3.3 of this paper) represent special cases of Theorem 3. In addition, Theorem 3 covers more cases for both single- and multi-output circuits.

4.3 Application of E/O Theory

First, let us show the algorithm that verifies the conditions in Theorem 2(1) and Theorem 3(1). Any combinational circuit can be represented as a DAG (Directed-Acyclic Graph) in which nodes correspond to AND, OR, or NOT gates. The directed edges in a DAG correspond to connections among gates. The in-coming edges to a node represent the connections with the inputs of the gate, and the out-going edge represents the output from the gate. More than one out-going edges from a node represent a fan-out.

Algorithm 1: Checking testability-preserving conditions for substitution

INPUT: The original circuit C_1 and candidate nodes B and C for substitution.

OUTPUT: True, if at least one of the conditions is satisfied; False, otherwise.

METHOD:

1. Extract the subgraph G_0 from the DAG of C_1 so that G_0 contains only all the paths from B and C to all the primary outputs $\{f_i\}$.
2. Mark each node in G_0 as P_B and/or P_C if it is on a path from B and/or C to a primary output.
3. Denote node set $N_{BC} = \{n | n$ is marked as both P_B and $P_C\}$.
4. IF N_{BC} is empty, THEN RETURN True.
5. Denote node set $N_p = \{n | n \in N_{BC}$, no node directly connected with the input of node n belongs to $N_{BC}\}$.
6. FOR each $n \in N_p$ DO
   (a) Calculate the *E/O* values of B and C, respectively.
   (b) IF conflict occurs THEN RETURN False.
   (c) IF C_1 is redundant, and B and C have opposite *E/O* values THEN RETURN False.
7. RETURN TRUE.

Based on Theorems 2 and 3, the following algorithm is proposed. This algorithm can perform general optimizations that are testability-preserving even if the circuits to be optimized are not fully testable. The basic idea of the algorithm is that in addition to the extraction of factorization, as that in [4], subfunctions in different levels are also substituted.

Algorithm 2: Factorization and substitution on redundant and irredundant circuits

INPUT: An initial circuit C_1 having a cost c_1.

OUTPUT: An optimized circuit C_2 having a cost c_2 such that $c_2 \leq c_1$, and C_2 preserves the testability of C_1.

METHOD:

Perform the testability-preserving optimization on the circuit until no further optimization can be made as follows.

1. Find factorization $Plan_{fact}$ with the maximum saving in the number of literals in the expression (or transistors in the circuit) using single-cube, double-cube, or dual expression extraction.
2. For each node pair x and y which represent the same Boolean function, check whether one of the testability-preserving conditions is satisfied by using Algorithm 1. Find substitution $Plan_{subst}$ with the maximum saving in the number of literals in the Boolean expression (or transistors in the circuit).
3. Compare the saving in the number of literals in the expression (or transistors in the circuit) of $Plan_{fact}$, and $Plan_{subst}$. If the savings are all 0, then exit; otherwise denote the plan with the maximum saving in the number of literals in the expression (or transistors in the circuit) as *Plan*.
4. Perform optimization *Plan*; go to Step (1).

TPOS can optimize circuits using both factorization and substitution as described above. It can also check the *E/O* values of the nodes to be optimized. If the option in TPOS which indicates optimization of redundant circuits is set, then only nodes with the same E/O values are eligible for optimization.

FAN [16] is a fast test generation system for VLSI circuits. In this paper, the fault-simulation part of FAN is used to measure the testability of circuits. It should be pointed out that the fault collapsing technique [12, 3] is used in FAN. This technique reduces the number of faults that need to be considered in order to completely test a given circuit. However, with fault collapsing, the fault simulator in FAN gives the number of faults after collapsing instead of the real number of existing faults. Therefore, in the experimental part of this paper, the fault collapsing part of FAN is disabled so that we can observe the actual number of undetectable faults in the circuits simulated.

5 EXPERIMENTAL RESULTS

5.1 Design of the Experiment

A program, named TPOS (Testability-Preserving Optimization System), which implements both Algorithm 1 and Algorithm 2 is coded in about 5000 lines of C and run under Unix on a Sun Sparc 2 workstation. The purpose of developing TPOS is mainly to verify the validity of the theory proposed in the paper.

5.2 Explanation of the Experimental Results

The program TPOS has been used to test twenty-five examples including the circuits used in [9] and the benchmarks distributed by the Microelectronics Center of North Carolina (including the ISCAS'85 set) [17, 18]. The test data and the experimental results are listed in Table I and Table II.

TABLE I The Irredundant Circuits Optimized by TPOS

Circuit name	Circuit function	# of Gates/# of lines		Undetectable/total faults	
		original	optimized	original	optimized
9symml	Count Ones	162/423	151/399	0/846	0/798
C17	Logic	17/28	15/26	0/56	0/52
cc	Logic	97/210	77/160	0/420	0/320
f4	SOP Logic	18/124	36/104	0/248	0/208
f51m	Arithmetic	90/425	106/303	0/850	0/606
f6	SOP Logic	19/84	25/67	0/168	0/134
lal	Logic	211/468	171/366	0/936	0/732
majority	Voter	13/36	13/26	0/72	0/52
my_adder	Adder	224/578	257/563	0/1156	0/1126
z4ml	2-bit Add	70/336	81/211	0/672	0/422

TABLE II The Redundant Circuits Optimized by TPOS

Circuit name	Circuit function	# of Gates/# of lines		Undetectable/total faults	
		original	optimized	original	optimized
b1	Logic	15/39	15/36	18/78	6/72
C1355	Logic	1186/1995	1154/1963	8/3990	8/3926
C432	Priority Decoder	412/711	374/681	53/1422	49/1362
c8	Logic	266/660	234/500	181/1320	29/1000
cht	Logic	161/586	189/469	160/1172	37/938
cm150a	Logic	110/224	96/180	59/448	15/360
cmb	Logic	55/138	51/124	14/276	2/248
cu	Logic	64/168	61/156	18/336	6/312
f13	SOP Logic	26/159	34/87	6/318	1/174
f5	SOP Logic	21/87	28/70	28/174	9/140
mux	MUX	87/252	97/209	117/504	48/418
sct	Logic	160/404	132/298	7/808	2/596
tcon	Logic	33/99	33/83	24/198	8/166
ttt2	Logic	508/1247	355/1110	708/2494	521/2220

The number of signal lines in a circuit is roughly equal to the number of transistors in the circuit. Minimizing the number of transistors in the circuit is used as an optimization criterion in TPOS. In the results listed in this paper, the number of the gates in some circuit may increase after optimization because the number of the gates is not selected as the optimization goal in TPOS.

Table I lists data for irredundant circuits optimized by TPOS. By comparing the last two columns of the table, it is clear that the optimized circuits are also irredundant. In other words, optimization preserves the testability for the irredundant circuits.

Table II lists data for redundant circuits optimized by TPOS. By comparing the number of undetectable faults of the original circuits and that of the optimized circuits, it is clear that the optimized circuits have the same or better testability than the original circuits. The types (stuck-at-0 or stuck-at-1) and the locations of undetectable faults in the original and the optimized circuits were compared (not listed in the table due to space limitation). It was found that the type and location of undetectable faults are the same except that some undetectable faults in the original circuit were removed because the lines on which these undetectable faults occur were removed.

Circuit optimization may affect the test generation so that the test generation program needs more CPU time for the optimized circuits, or it may need more test vectors to test the optimized circuits. These problems are not considered in the proposed theory. However, by comparing the data for test generation listed in Table III, the average CPU time cost for FAN to

TABLE III Test Generation for the Circuits Optimized

Circuit name	CPU time costed by FAN (sec)		Number of test patterns needed	
	original	optimized	original	optimized
9symml	1.02	0.81	133	118
C17	0.02	0.02	7	7
cc	0.07	0.05	29	21
f4	0.13	0.10	75	52
f51m	0.60	0.23	94	50
f6	0.07	0.05	47	35
lal	0.55	0.27	88	65
majority	0.02	0.00	14	9
my_adder	0.23	0.25	20	38
z4ml	0.37	0.12	73	40
b1	0.00	0.00	4	5
C1355	15.47	15.28	147	147
C432	537.72	555.5	87	84
c8	0.93	0.42	77	70
cht	0.72	0.17	55	30
cm150a	0.33	0.22	59	56
cmb	0.13	0.08	37	33
cu	0.13	0.10	48	45
f13	0.30	0.05	98	35
f5	0.07	0.05	28	24
mux	0.50	0.30	89	71
sct	0.40	0.25	79	61
tcon	0.02	0.02	11	11
ttt2	4.75	1.42	127	99

generate test patterns and the average number of test patterns needed drop after the circuits were optimized with testability preservation. In other words, because the scales of circuits (i.e., the number of components and the number of signal lines) are optimized, CPU time for test pattern generation can be saved, and it generally needs fewer test patterns to test smaller or simpler circuits.

6 CONCLUSIONS AND OPEN RESEARCH TOPICS

In this paper, a new theory for testability-preserving optimization of combinational circuits is established. According to this theory, in a redundant or irredundant circuit, substitutions which satisfy one of the conditions described in part 1 of Theorems 2 and 3, double-cube factorization, and extraction of NOT gates preserve testability, no matter whether the circuits have single- or multiple-output.

The theory proposed allows us to design a more generalized optimization algorithm which preserves testability for both redundant and irredundant circuits. This algorithm uses both factorization and substitution methods. The experimental results reported show that both irredundant and redundant circuits preserve testability after being optimized using the proposed algorithm.

There are some improvements to be done on the theory and the algorithm proposed in this paper. For example, the theory may be generalized to be applicable to a functional block level model.

The fact that circuit optimization may affect the test generation is not considered in the proposed theory. The theory only guarantees that no new undetectable faults in the optimized circuit are created, but it cannot guarantee that the cost of testing and test generation does not increase. In the context of DFT (Design For Testing) [3] [19], testability is defined as the ease of testing or as the ability to test easily or cost-effectively [19]. To preserve the testability in the context of DFT during optimizing is another open research topic to be studied.

References

[1] M. Wu, W. Shu, and S. Chan, "A unified theory for MOS circuit design-switching network logic," *Int. J. Electronics*, Vol. 58, No. 1, 1985, pp. 1–33.

[2] B. W. Johnson, *Design and analysis of fault tolerant digital systems*, Addison-Wesley Publishing Company, 1989.

[3] D. K. Pradhan, *Fault-tolerant computing*, Vol. 1, Prentice-Hall, 1986.

[4] J. Rajski, and J. Vasudevamurthy, "The testability-preserving concurrent decomposition and factorization of Boolean expressions," *IEEE Trans. on CAD*, Vol. 11, No. 6, June 1992, pp. 778–793.

[5] D. L. Dietmeyer, and Y. H. Su, "Logic design automation of fan-in limited NAND networks," *IEEE Trans. on Comput.*, Vol. C-18, No. 1, Jan. 1969, pp. 11–22.

[6] R. K. Brayton, and C. MucMullen, "Synthesis and optimization of multistage logic," *Proc. Int. Conf. Computer Design*, 1984, pp. 23–28.

[7] R. K. Brayton, R. Rudell, A. Sangiovanni-Vincentellim, and A. Wang, "Multilevel logic optimization and the rectangular covering problem," *Proc. of ICCAD-87*, Nov. 1987, pp. 66–69.

[8] R. K. Brayton, "Factoring logic functions," *IBM J. Res. & Develop.*, Vol. 31, No. 2, Mar. 1987, pp. 187–198.

[9] G. Caruso, "Near optimal factorization of Boolean functions," *IEEE Trans. on CAD*, Vol. 10, No. 8, Aug. 1991, pp. 1072–1078.

[10] R. K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. R. Wang, "MIS: A multiple-level logic optimization system," *IEEE Trans. on CAD*, Vol. CAD-6, No. 6, Nov. 1987, pp. 1062–1081.

[11] Z. Kohavi, *Switching and finite automata theory*, McGraw-Hill, 1978.

[12] N. K. Jha, and S. Kundu, *Testing and reliable design of CMOS circuits*, Kluwer Academic Publishers, 1990.

[13] G. Hachtel, R. Jacoby, K. Keutzer, and C. Morrison, "On Properties of Algebraic Transformations and the Multifault Testability of Multilevel Logic," *Proc. Int. Computer Aided Design*, 1989, pp. 422–425.

[14] J. P. Roth, "Diagnosis of automata failures: A calculus and a method," *IBM J. Res. & Develop.*, Vol. 10, July 1966, pp. 278–291.

[15] E. J. McCluskey, *Logic design principles with emphasis on testable semicustom circuits*, Prentice-Hall, 1986.

[16] H. Fujiwara, "II-331 FAN: a fast test generation system for VLSI circuits," *Research report of Engineering Institute, Meiji University*, No. 50, March 1986, Meiji University, Japan.

[17] E. M. Sentovich, et al., "SIS: a system for sequential circuit synthesis," *Research report*, Momorandum No. UCB/ERL M92/41, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, CA. 94720, May 1992.

[18] S. Yang, "Logic synthesis and optimization benchmarks user guide," *Research report*, Microelectronics Center of

North Carolina, P.O. Box 12889, Research Triangle Park, NC 27709, Jan. 1991.

[19] H. Fujiwara, *Logic testing and design for testability*, The MIT Press, 1985.

APPENDIX A: PROOF OF LEMMA 4

In order to prove Lemma 4, we need to prove Lemmas 10–15 first.

LEMMA 10 *Suppose that there is a single-path from A to x_1 in the circuits shown in Figure 11(a)–(c). Then we have:*

1. *If* $\frac{\partial x_1}{\partial A} \in \{0, 1\}$, *then* $\frac{\partial f_{OR}}{\partial A} \in \{0, 1\}$; *if* $\frac{\partial x_1}{\partial A} \in \{0, -1\}$, *then* $\frac{\partial f_{OR}}{\partial A} \in \{0, -1\}$.
2. *If* $\frac{\partial x_1}{\partial A} \in \{0, 1\}$, *then* $\frac{\partial f_{AND}}{\partial A} \in \{0, 1\}$; *if* $\frac{\partial x_1}{\partial A} \in \{0, -1\}$, *then* $\frac{\partial f_{AND}}{\partial A} \in \{0, -1\}$.
3. *If* $\frac{\partial x_1}{\partial A} \in \{0, 1\}$, *then* $\frac{\partial f_{NOT}}{\partial A} \in \{0, -1\}$; *if* $\frac{\partial x_1}{\partial A} \in \{0, -1\}$, *then* $\frac{\partial f_{NOT}}{\partial A} \in \{0, 1\}$.

Proof Because $\frac{\partial f_{OR}}{\partial x_1} \in \{0, 1\}$ and $\frac{\partial f_{AND}}{\partial x_1} \in \{0, 1\}$, from Lemma 3 we know that statements (1)–(3) are true. Q.E.D.

LEMMA 11 *Suppose that there is only one path from point A to f in the circuit in Figure 16. Then the following statements are true.*

(1) If $A \to^{E} f$, *then* $\frac{\partial f}{\partial A} \in \{0, 1\}$.

(2) If $A \to^{O} f$, *then* $\frac{\partial f}{\partial A} \in \{0, -1\}$.

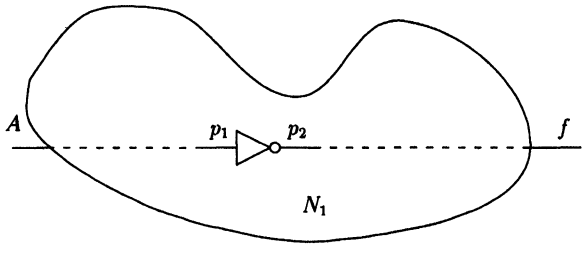


FIGURE 16 Illustration for Lemma 11

Proof If there is no inverter from A to f, from Lemma 2 and Lemma 10 we know that $\partial f / \partial A \in \{0, 1\}$.

Suppose that there is one inverter in the path from A to f, as shown in Figure 16. Let p_1 denote the input of the inverter, p_2 denotes the output of the inverter. Then, because $\frac{\partial p_2}{\partial A} = -\frac{\partial p_1}{\partial A} \in \{0, -1\}$, and there is no inverter from p_2 to f, so $\frac{\partial f}{\partial A} \in \{0, 1\}$.

In the same way, we can prove that if there are even number of inverters in the path from A to f, then $\frac{\partial f}{\partial A} \in \{0, 1\}$. Similarly, if there are odd number of inverters in the path from A to f, then $\frac{\partial f}{\partial A} \in \{0, -1\}$. Q.E.D.

LEMMA 12 *Let x_i and f denote respectively the input and output of an n-input OR (or AND) gate, where i = 1, 2,..., n, and A is a point in the circuit. Then the following statements are true:*

(1) If $\frac{\partial x_i}{\partial A} \in \{0, 1\}$ *for all* $i = 1,2,\ldots,n$, *then* $\frac{\partial f}{\partial A} \in \{0, 1\}$

(2) If $\frac{\partial x_i}{\partial A} \in \{0, -1\}$ *for all* $i = 1,2,\ldots, n$, *then* $\frac{\partial f}{\partial A} \in \{0, -1\}$

Proof Because an OR gate is monotonically increasing, when some of the inputs change from 0 to 1 (or 1 to 0) due to fault(s), f can only change from 0 to 1 (or 1 to 0), or remain unchanged. In other words, f can only change in the same direction as the inputs or remain unchanged.

The statements for the AND gate can be proven in the same way. Q.E.D.

Lemma 10 can be generalized to multi pathes case as shown in Lemma 13.

LEMMA 13 *Suppose that line x and p are any two lines in a circuit. Then we have:*

1. $\frac{\partial p}{\partial x} \in \{0, 1\}$, *if* $x \overset{E}{\to} p$;
2. $\frac{\partial p}{\partial x} \in \{-1, 0\}$, *if* $x \overset{O}{\to} p$;

3. $\frac{\partial p}{\partial x} = 0$, *if there is no path from x to p.*

Proof We proof this lemma using contradiction.

Suppose in a complex path from x to p, $x \xrightarrow{E} p$ while $\frac{\partial p}{\partial x} = -1$. Therefore, there is at least one input y to the gate whose output is p such that $\frac{\partial y}{\partial x} = -1$ if the gate p is an AND or OR gate, i.e., $x \xrightarrow{E} y$; or $\frac{\partial y}{\partial x} = 1$ if the gate p is NOT gate, i.e., $x \xrightarrow{O} y$.

By repeating the above reduction, we can eventually get a single path from x to z such that $x \xrightarrow{E} z$ and $\frac{\partial z}{\partial x} = -1$, or $x \xrightarrow{O} z$ and $\frac{\partial z}{\partial x} = 1$. This result conflicts with Lemma 11.

This proves the lemma. Q.E.D.

LEMMA 14 *Suppose that in a given circuit,* $B \xrightarrow{E} f$, *and* $C \xrightarrow{E} f$, *where f is the primary output, and nodes B and C represent the same Boolean function. The merge point M from B and C in the extracted circuit is testable, if B is testable in the original circuit.*

Proof Suppose that $B = D$ in the original circuit is detectable with test pattern t_1, i.e., $\frac{\partial f}{\partial B}|_{t_1} = 1$. That is, when B changes from 1 to 0, f also changes from 1 to 0.

From Lemma 13, $\frac{\partial f}{\partial B} \in \{0, 1\}$for any input. Therefore, when C changes from 1 to 0, f will not change from 0 to 1. Instead, f remains 0. So we have

$$f(M = D)|_{t_1} = f(B = D \cap C = D)|_{t_1} = D$$

Similarly, it can be shown that $M = \bar{D}$ is testable if $B = \bar{D}$ in the original circuit is testable. Q.E.D.

In the same way, we can prove the following lemma.

LEMMA 15 *Suppose that in a circuit* $B \xrightarrow{O} f$, *and* $C \xrightarrow{O} f$, *where f is the primary output, and nodes B and C represent the same Boolean function. The merge point M from B and C in the extracted circuit is testable, if B is testable in the original circuit.*

Now Lemma 4 can be proven.

LEMMA 4 *In the circuit shown in Figure 12, suppose that the E/O values of nodes B and C are the same. Then the extraction from Figure 12(a) to Figure 12(b) preserves testability.*

Proof Because $N_2 \backslash K_2$ ("\" represents graph difference) in the substituted circuit is unchanged, from Lemma 1, we know that $N_2 \backslash K_2$ preserves testability.

Now, let us consider the faults in sub-circuit K_2. Any of these faults can be represented by a symbol D on M in the substituted circuit. From Lemmas 14 and 15 we know that M is testable if B is testable in the original circuit. Hence, all faults in sub-circuit K_2 are detectable.

Therefore, the extraction preserves testability. Q.E.D.

APPENDIX B: PROOF OF LEMMA 8

LEMMA 8 *Consider the substitution in Figure 12. Suppose that all paths from B and C to f meet only at f, and* $B \xrightarrow{E} f$, *and* $C \xrightarrow{O} f$. *The substitution preserves testability, if at least one of the following conditions is satisfied:*

1. *Gate f is an AND gate, and single faults* $B = D$ *and* $C = \bar{D}$ *in the original circuit are detectable.*
2. *Gate f is an OR gate, and single faults* $B = \bar{D}$ *and* $C = D$ *in the original circuit are detectable.*

Proof What is needed is to prove that all single faults which are represented by $M = D$ or $M = \bar{D}$ in the extracted circuit can be detected.

(1) Case 1: Because single fault $B = D$ in the original circuit is detectable, i.e., there exists a test vector t_1 such that

$$f(B = D)|_{t_1} = D.$$

Under fault-free condition, $B = 1$ and $C = 1$. When $B = D$ and under the test vector t_1, $x_1 = D$ and $x_2 = 1$ so that fault $B = D$ can propagate (otherwise,

$x_2 = 0$ will conclude that $B = D$ cannot propagate to f, i.e., is not detectable. The conflict proves that $x_2 = 1$.)

Now, let C change from 1 to 0 such that the situation of the circuit is equivalent to the effect of fault $M = D$. Because $C \xrightarrow{O} f$, i.e., $C \xrightarrow{O} x_2$, from Lemma 13, we know that $\frac{\partial x_2}{\partial C} \in \{0, -1\}$. Therefore, x_2 will not change from 1 to 0, but will remain at 1. Hence,

$$f(\mathrm{M} = \mathrm{D})|_{t_1} = f(\mathrm{B} = \mathrm{D} \cap \mathrm{C} = \mathrm{D})|_{t_1} = \mathrm{D}$$

Therefore, any fault which can be represented as $M = D$ is detectable.

Similarly, from $C = \bar{D}$, it can be shown that $M = \bar{D}$ is detectable.

(2) The proof is similar to case (1). Q.E.D.

Authors' Biographies

Jiabi J. Zhu received the B. Eng. degree in computer science and engineering, and M. Eng. degree in Electrical Enginerring from Tsinghua University, Beijing, China in 1985 and 1990, respectively, and the M.Sc. degree in Computer Science from University of Saskatchewan, Saskatoon, SK, Canada, in 1992.

Following his graduate studies, he joined SED Systems, Inc., in Saskatoon, Canada, designing products of telecommunication protocol conversion. Since 1994, he joined Silicon Valley Research, Inc., Mountain View, California where he is holding a senior position. His work there is in the area of placement algorithm and VLSI layout methodologies.

His interests are in VLSI layout, fault tolerant computing, and circuit optimization.

Mostafa H. Abd-El-Barr (S'85-M'85-SM'92) received the B.Sc. and M.Sc. degree in Electrical Enginerring in 1973 and 1979, respectively, from Cario University, Egypt, and the Ph.D. degree in Electrical Enginerring from the University of Toronto, Ont., Canada in 1986.

In July 1986 he joined the Faculty of the Department of Computer Science, University of Saskatchewan, Saskatoon, Canada, where he is currently a Full Professor. He is also an Associate Member of the Department of Electrical Enginerring at the same university. He teaches courses on digital system design, computer hardware and architecture, VLSI systems design methodology, fault-tolerant computing, and CMOS testability. His research interests include VLSI design and implementation of algorithms, CMOS testability, fault-tolerant computing, and multiple-valued logic systems design. He is the author and/or coauthor of about 50 scientific papers published in journals and conference proceedings.

Carl McCrosky received a PhD (1985) in Electrical Engineering and an MSc in Computer Science from Queen's University, Kingston, Canada. He received a BA from the Centre College of Kentucky in Danville.

McCrosky is a Professor of Computer Science at the University of Saskatchewan since where he has taught since 1985. He is also an Adjunct Professor at TR Labs, Saskatoon. McCrosky was a founding partner of Andyne Computing Ltd, Kingston, Canada.

His research considers three areas: bandwidth-on-demand telecommunications systems, synthesis of VLSI systems, and first-class functional array languages.