LOT: Logic Optimization with Testability—New Transformations for Logic Synthesis

Mitrajit Chatterjee, Dhiraj K. Pradhan, *Fellow, IEEE,* and Wolfgang Kunz, *Member, IEEE*

***Abstract*—A new approach to optimize multilevel logic circuits is introduced. Given a multilevel circuit, the synthesis method optimizes its area while simultaneously enhancing its random pattern testability. The method is based on structural transformations at the gate level. New transformations involving EX-OR gates as well as Reed–Muller expansions have been introduced in the synthesis of multilevel circuits. This method is augmented with transformations that specifically enhance random-pattern testability while reducing the area. Testability enhancement is an integral part of our synthesis methodology. Experimental results show that the proposed methodology not only can achieve lower area than other similar tools, but that it achieves better testability compared to available testability enhancement tools such as *tstfx*. Specifically for ISCAS-85 benchmark circuits, it was observed that EX-OR gate-based transformations successfully contributed toward generating smaller circuits compared to other state-of-the-art logic optimization tools.**

***Index Terms*— ATPG, built-in self-test, logic synthesis, optimization, testability.**

I. Introduction

MOST state-of-the-art synthesis and optimization tools on multilevel combinational circuits focus either on area [4], [8], [9], [15], performance [17], power [38], or testability [28]. Targeting multiple design requirements in an integrated framework is becoming increasingly useful. Although EX-OR gate-based synthesis of two or three-level logic and Reed–Muller expansions [29], [31] have been explored, not much work has been reported on general multilevel logic synthesis using EX-OR gates. Efficient usage of EX-OR gates in multilevel circuits can be advantageous to a variety of circuits. These EX-OR gates not only can reduce the circuit area, but they also make the circuit more easily testable [31], [34]. In this paper, synthesis is done on multilevel circuits with the objective of area optimization as well as testability enhancement. Transformations done during synthesis include new transformations introducing EX-OR gates in multilevel circuits. Our synthesis method, Logic Optimization with Testability (LOT), can target effectively both area optimization as well as testability—thus integrating the multiple design goals in an uniform framework. It is important to note that transformations [4], [8] using indirect implications [1] provide a powerful, unified theory for the entire synthesis process [12]. Further extensions of this framework to delay optimization as well are currently underway.

The synthesis procedure is based on the gate-level description of a multilevel circuit. The logic optimization with Boolean transformations at the structural level [4], [8], [9], [14] can be more memory efficient [4], [8] and closer to the physical reality of the design compared to a functional level logic optimization based on Boolean networks or binary decision diagrams (BDD's) [15], [17]. Furthermore, working on the gate-level can give a better view of other design costs during synthesis. It applies logic transformations to a multilevel circuit to yield reduced area as well as enhanced random testability. The transformations used here are of two types:

i) transformations based on indirect implications;
ii) new transformations using EX-OR gates as well as Reed–Muller expansions.

Recursive learning techniques [1]–[4] form the basic mathematical underpinning of our synthesis tool. The transformations of the first type make use of efficient redundancy identification techniques to make circuit transformations. These transformations can perform a wide range of manipulations in a combinational network [4], [8], thus covering a large design space. The transformations of the second type are based on a) replacing conventional gates with EX-OR gates, b) introducing EX-OR gates, and c) introducing EX-OR gates through Reed–Muller transformations. Using EX-OR gates as primitive gates in synthesis has drawn recent attention [29], [30], [34], particularly in implementing arithmetic and linear functions, telecommunication, encryption, and encoding schemes [29]. Recent designs of programmable logic designs (PLD) and field-programmable gate arrays (FPGA's) also include EX-OR gates in their logic units. Moreover, circuits with more EX-OR gates tend to be more testable [31], [34]. Transformations of both types can be used iteratively to yield an optimized circuit.

While most of the synthesis tools ensure that all faults of interest are testable, random testability [32] of the faults, a desirable requirement of built-in-self-test (BIST), is not considered. Synthesis techniques presented to date [18], [20], [21] for random pattern testable circuits require two-level circuit descriptions as inputs. Our technique presented here differs in that it can take any multilevel circuit as input to

Manuscript received February 26, 1996; revised April 9, 1997. This research was supported in part by NSF Grant MIP 9406946 and ONR Grant N00014-92-J-1366. This paper was recommended by Associate Editor K. T. Cheng.

M. Chatterjee is with Design Automation Group, Integrated Device Technology Inc., Santa Clara, CA 95054 USA.

D. K. Pradhan is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305-9020 USA, on leave from the Department of Computer Science, Texas A&M University, College Station, TX 77843 USA.

W. Kunz is with the Department of Computer Science, University of Frankfurt, Frankfurt, Germany.

Publisher Item Identifier S 0278-0070(98)04627-2.

the synthesis tool. Thus, this methodology can not only be applicable in an environment of engineering changes, but also be used as a postprocessor for other synthesis/optimization tools. The area optimization results presented here are the best reported thus far.

The paper is organized as follows. The next section provides a brief summary of previous work. Section III outlines some preliminaries and reviews logic synthesis using implication-based Boolean (IB) transformations. Section IV introduces new transformations based on EX-OR gates. Section V presents synthesis for area optimization and its performance in ISCAS-85 benchmark circuits. Section VI describes an algorithm for random pattern testability enhancement, presenting related experimental results. We conclude in Section VII.

II. Previous Work

A survey of multilevel combinational logic optimization techniques based on Boolean networks can be found in [17]. Logic synthesis based on structural transformations has been effectively applied in [4], [8], [9], and [14]. Synthesis for area optimization was done by transformations [8], [9] using redundancy elimination based on adding and removing connections in the circuit. These transformations were also applied in the form of permissible bridges after technology mapping in [14]. A generalized form of the IB transformations, based on orthonormal expansions, has been presented in [4]. The *global flow* method proposed by Berman and Trevillyan [17], [42] iteratively uses two steps in the network: a) making a connection based on an implication in the network (reduction) and b) change other nodes based on the new transformation (expansion). Though the method exploits global transformations and the Boolean model, it only uses satisfiability don't cares in the network.

Logic synthesis, with extensive use of EX-OR gates, has been limited to two-level (AND-EXOR) and three-level (AND-OR-EXOR) designs (see [29]). These methods develop algorithms for optimizing exclusive-or sum-of-products, yielding a minimum number of product terms. Earlier, [31], [34], and [40] have shown that EX-OR gates enhance testability for AND-EXOR designs.

Logic synthesis with testability enhancement has been studied at various levels of synthesis—starting from high-level synthesis to technology mapping [28]. In the context of random testable logic synthesis of the unmapped circuit, previous research consists of redundancy removal from combinational circuits [17], synthesis of fully testable circuits [13], [23], testability-preserving of multiple stuck-at-faults, path delay faults [28], [41], and synthesis of random testable circuits from two-level circuits [20], [21].

It has been proposed in [18] that careful assignment of don't cares of functions and redundant lines can improve the detectability profile of circuits. The two previous random pattern testable circuit synthesis methods [20], [21] start with a two-level circuit, first determining the random testability of the circuit and identifying the hard faults. Multilevel circuits are synthesized from two-level circuits using transformations, with each step being evaluated based on its impact on the random testability of the circuits. The transformations in [20] are limited to algebraic factors, namely kernels and common cubes. The algebraic transformations used in [21] include *single cube division*, *double cube division*, and some *double cube divisions with multiple outputs*. The synthesis procedure in [20] also uses testability-preserving transformations proposed in [13] and [23] and inserts test points based on fault detection probabilities.

III. Transformations for Synthesis

The proposed synthesis method takes as its input a multilevel combinational circuit and optimizes area while enhancing testability. Therefore, the cost function used can include both area and random pattern testability. Structural transformations are used to minimize the cost function. This section gives the basic preliminaries for synthesis and presents the transformations to be used here.

A. Preliminaries

Assume a combinational circuit C is given with n primary inputs and m primary outputs. The combinational circuit consists of primary gates like AND, OR, NOT, NAND, NOR, and EX-OR gates. All gates in the circuit have a unique label, and their output signals y_i realize Boolean functions $y_i(\vec{x})$: $B_2^n \rightarrow B_2$ with $B_2 = \{0, 1\}$, where the variable $x_1, \cdots, x_n$ corresponds to the primary input signals of the circuit C. Following the usual representation of a combinational circuit as a directed acyclic graph (DAG), a signal f lies in the transitive fanout of y if and only if there exists a directed path from y to f. Furthermore, we assume that there are no external don't cares; the function of the combinational network $C(\vec{x})$: $B_2^n \rightarrow B_2^m$ with $B_2 = \{0, 1\}$ is completely specified.

Two combinational networks, C and C', are called equivalent, denoted as $C = C'$ if they implement the same function, $C(\vec{x})$: $B_2^n \rightarrow B_2^m$ with $B_2 = \{0, 1\}$. They are called structurally identical [1] if there exists a one-to-one mapping between C and C', such that for every node y_i in C, there is a y_i' in C' and vice-versa, where y_i and y_i' implement the same function. A transformation from a network C to another equivalent network C' is possible by replacing the node y in C by a node y' with an equivalent function. A function at node y can also be replaced by some nonequivalent function y' if this does not change the function $C(\vec{x})$: $B_2^n \rightarrow B_2^m$ of the logic network as a whole. Such functions are called permissible functions [11].

B. Implication-Based Boolean Transformations—Review

Indirect implications derived by using recursive learning have been shown to be useful in testing verification and optimization [1]–[4]. Given a set of value assignments, recursive learning can be used to obtain all indirect implications. The procedure allows the user to set the maximum level of recursion to control the computational time. Therefore, this provides a way of trading off between time for transformations and redundancy removal. In this paper, we rely on Boolean transformations derived by using indirect implications. These

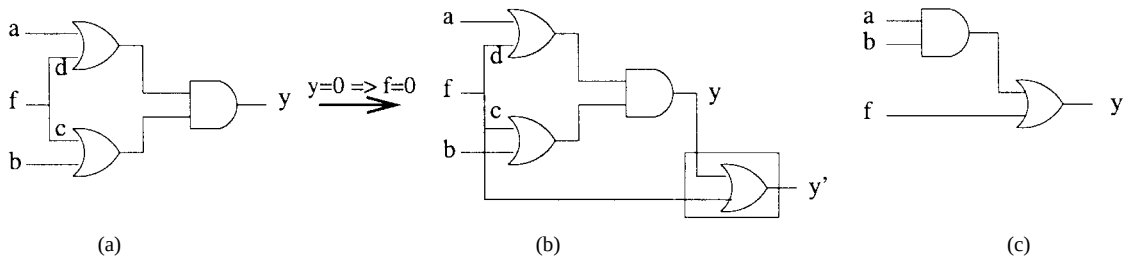

Fig. 1. An example of an IB transformation.

transformations can be referred to here as IB transformations. They have three main advantages.

- They are simple, and the cost of each transformation can be estimated with low time overhead.
- These transformations preserve the functionality of the circuit; therefore, there is no need to verify for equivalence after transformation.
- The expansions based on IB transformations can cover a wide variety of logic transformations [4], [8].

The following reviews the basic transformations already presented in [4]. We augment these transformations with new ones for our tool. The factorization techniques commonly used in multilevel optimization can be derived based on the expansion

$$y(\vec{x}) = f(\vec{x}).y(\vec{x})\big|_{f(\vec{x})=1} + \overline{f}(\vec{x}).y(\vec{x})\big|_{f(\vec{x})=0}. \qquad (1)$$

Here, we use the following notation, $y = f.y|_1 + \overline{f}.y|_0$, to represent the above equation. The above expression can take the form of a division operation with $f(\vec{x})$ as the divisor, $y(\vec{x})$ as the quotient, $y(\vec{x})|_{f(\vec{x})=1}$ as the dividend, and $\overline{f}(\vec{x}).y(\vec{x})|_{f(\vec{x})=0}$ as the remainder. The terms $y(\vec{x})|_{f(\vec{x})=1}$ and $y(\vec{x})|_{f(\vec{x})=0}$ denote the cofactors of this expansion. The above expression can be interpreted as a generalized form of the well-known Shannon expansion [4]. The main issue of this approach is to divide [15], [17] function $y(\vec{x})$ by appropriate divisors $f(\vec{x})$, such that the exploitation of the internally created don't cares results in a reduced circuit. Though don't care conditions are not explicitly calculated, a test pattern generation tool is used to remove the redundancies resulting from these don't cares. This leads to the following two-step process.

1) Transformation: $y = f.y + \overline{f}.y$.
2) Reduction: Redundancy elimination.

The above can be seen as a special way of performing a Boolean division [17] for some dividend y and some divisor f (Boolean division is not unique). The method to identify divisors is based on indirect implications [1], [2]. If a value assignment at a node y allows us to imply a unique value assignment at node f, then the four transformations in Table I are valid [4]. The node f must not be in the transitive fanout of y to ensure that the circuit remains combinational after the transformation.

Example 3.1: Take the circuit shown in Fig. 1(a) as an example. It can be observed that $y = 0 \Rightarrow f = 0$ and a transformation based on Condition 2) can be applied on the circuit. The resultant circuit is shown in Fig. 1(b). Redundancy elimination on the resultant circuit would determine lines "c" and "d" to be as redundant stuck-at-zero (s-a-0) lines and, hence, can be eliminated. The final circuit after redundancy elimination is shown in Fig. 1(c). □

TABLE I
TRANSFORMATIONS BASED ON IMPLICATIONS

	Condition	*Transformations*
1	y=0 ⇒ f=1	y'= $\bar{f}$ + y$\|_1$
2	y=0 ⇒ f=0	y'= f + y$\|_0$
3	y=1 ⇒ f=1	y'= f.y$\|_1$
4	y=1 ⇒ f=0	y'= $\bar{f}$.y$\|_0$

Transformations identified by testability-based conditions [4] are related to permissible functions.

Definition 3.1: Given a line y and a function f, we say that there is a detectability condition if $f = U$, $U \in 0, 1$ is a value assignment which is necessary to detect the fault, ys-a-$\overline{V}$, $V \in 0, 1$.

The above definition is a definition of condition for detectability (CD) relation identifying a relationship between a line and a function. The relation has been referred to as D-implications in [4]. A CD relation essentially identifies a necessary condition, $f = U$, to detect a change in the value on node y from V to $\overline{V}$. In other words, if $f = \overline{U}$, then the fault ys-a-$\overline{V}$ becomes undetectable. This is denoted as

$$y = V \overset{\text{CD}}{\rightarrow} f = U.$$

This can also be extended to a multiway relation in the circuit. The conventional implications can be viewed as a special case of such CD relations. These relations can exploit the controllability as well as observability conditions in the concerned nodes. The transformations of [4] are formulated below in terms of the CD relations.

Theorem 3.1: Let f and y be arbitrary nodes in a combinational network, C, where f is not in the transitive fanout of y, and y is an irredundant node in the network. The following permissible functions are possible.

a) The function y' with $y' = y|_1 + \overline{f}$ is a permissible function at node y if and only if the CD relation $y = 0 \overset{\text{CD}}{\rightarrow} f = 1$ is true.
b) The function y' with $y' = y|_0 + f$ is a permissible function at node y if and only if the CD relation $y = 0 \overset{\text{CD}}{\rightarrow} f = 0$ is true.
c) The function y' with $y' = y|_1.f$ is a permissible function at node y if and only if the CD relation $y = 1 \overset{\text{CD}}{\rightarrow} f = 1$ is true.

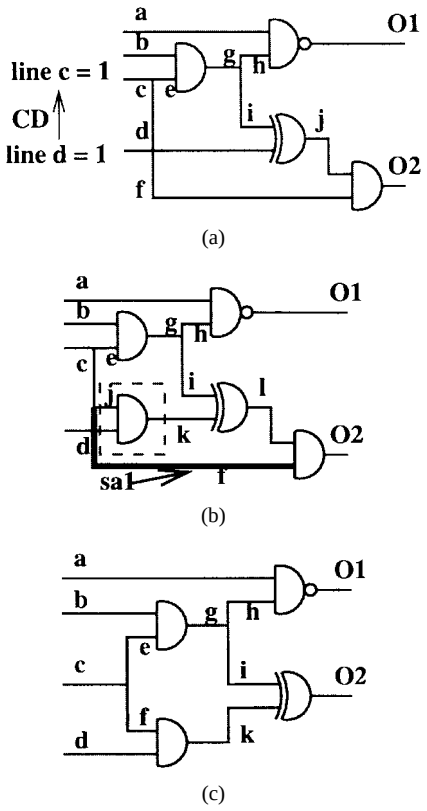

Fig. 2. An example of an IB transformation using CD relations.

d) The function y' with $y' = y|_0.\overline{f}$ is a permissible function at node y if and only if the CD relation $y = 1 \overset{\text{CD}}{\rightarrow} f = 0$ is true. □

In this paper, Transformations 1–4 will refer to the above permissible changes, a)–d), respectively, followed by redundancy removal.

Example 3.2: The circuit shown in Fig. 2(a) has a CD relation line d $= 1 \overset{\text{CD}}{\rightarrow}$ line c $= 1$. This implies that for line d $= 1$ to be observable at any one of the primary outputs, line c has to be set to the value one. Thus, Transformation 3 can be applied on the circuit. The resultant circuit is shown in Fig. 2(b). Redundancy elimination on the resultant circuit would determine line f to be stuck-at-one (s-a-1) redundant. The final circuit after redundancy elimination is shown in Fig. 2(c). □

CD relations to be used in the transformations are determined by recursive learning. This is accomplished by two routines: *make_all_implications()* and *fault_propagation_learning()*, as given in [1], if they are performed to the Roth's five-valued logic alphabet. The implication-based synthesis procedure only looks at indirect implications as the promising candidates for optimization.

IV. Transformations with EX-OR Gates

Circuits transformations based on Transformations 1–4 can make manipulations in a combinational network based on AND, OR, and NOT gates. These transformations, however, preclude EX-OR gates and other complex gates. Transformations involving EX-OR gates and complex gates can provide an added dimension in the search for circuit optimization. This is evident from the results presented later. New implication-based transformations introduce EX-OR gates a) by replacing a two-input gate with an EX-OR or EX-NOR gate, b) addition of an extra EX-OR gate, and c) using Reed–Muller Expansions. It may be noted that, as in [4], these transformations are designed to preserve functionality, thus avoiding the expensive step of checking the validation of the transformation. Also, they introduce redundancy in the circuit which, when removed, can often lead to a reduced circuit.

A. Transforming Gate Functionality

Two-input gates in circuits can be transformed to EX-OR gates or EX-NOR gates if they are *permissible*, based on implications [9], [11], [17]. A two-input gate, however, when transformed to an EX-OR gate, does not necessarily enhance the random testability. If the transformation is followed by other transformations, as we show, the EX-OR gate may enhance random pattern testability. A two-input AND (OR) gate or a two-input NOR (NAND) gate can be transformed to an EX-NOR (EX-OR) gate, due to an observability don't care condition [9], [17]. We can identify two such types of gate transformations.

Theorem 4.1: Let y be the output of an AND (NOR) gate with two inputs, a_0 and a_1. If the condition $y = 0 \overset{\text{CD}}{\rightarrow} (a_0 \oplus a_1 = 1)$ holds, this gate can be replaced by a permissible EX-NOR gate, now y becoming equal to $\overline{(a_0 \oplus a_1)}$.

Proof: The value assignments at the input of an AND (NOR) gate required to test a s-a-1 at the output of the gate are $\{0, 0\}$, $(\{1, 1\})$, $\{0, 1\}$, and $\{1, 0\}$. When the necessary condition to test the s-a-1 fault satisfies the relation $(a_0 \oplus a_1 = 1)$, it implies that only possible value assignments that can test the fault are $\{0, 1\}$ and $\{1, 0\}$. The value assignment $\{0, 0\}(\{1, 1\})$ to inputs of the AND (NOR) gate is either not satisfiable or observable, and hence, can be treated as a don't care to the circuit behavior. Thus, under such a condition, transforming the AND (NOR) gate to an EX-NOR gate does not change the functionality of the circuit, and is permissible. □

Theorem 4.2: Let y be the output of an OR (NAND) gate with two inputs, a_0 and a_1. If the condition $y = 1 \overset{\text{CD}}{\rightarrow} (a_0 \oplus a_1 = 1)$ holds, this gate can be replaced by a permissible EX-OR gate, now y becoming equal to $(a_0 \oplus a_1)$.

Transformations 5 and 6: Transformations based on Theorems 4.1 and 4.2 followed by redundancy removal.

Example 4.1: The circuit shown in Fig. 3(a) can be represented as (ab + bc + ca). At node d, the condition for a change from one to zero, to be observed at c, is $[\{a_1, a_0\} = \{0, 1\}, \{a_1, a_0\} = \{1, 0\}]$. This condition can allow the two-input OR gate at node d to be transformed to an EX-OR gate, (Transformation 6), yielding the circuit in Fig. 3(b). □

The search for such gate functionality transformations can be used in conjunction with other IB transformations. What these transformations will provide is an added flexibility in the search for optimal area.

B. Transforming with Addition of EX-OR Gates

In the above, we described how one can replace a two-input gate with an EX-OR gate. The following shows how one

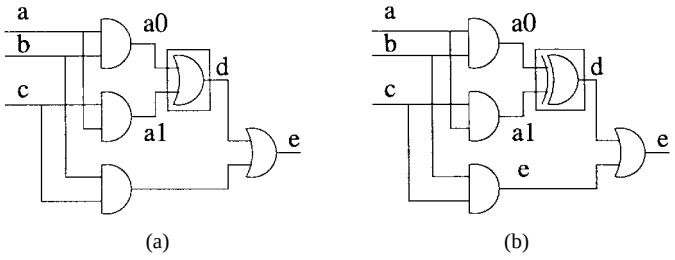

Fig. 3. An example of Transformation 6.

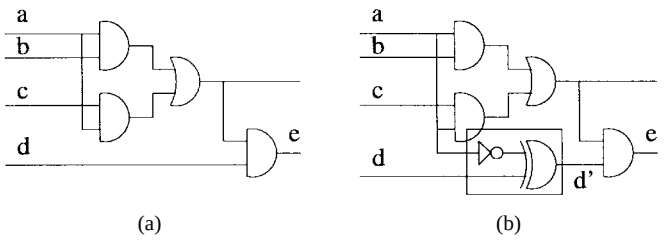

Fig. 4. An example of Transformation 8.

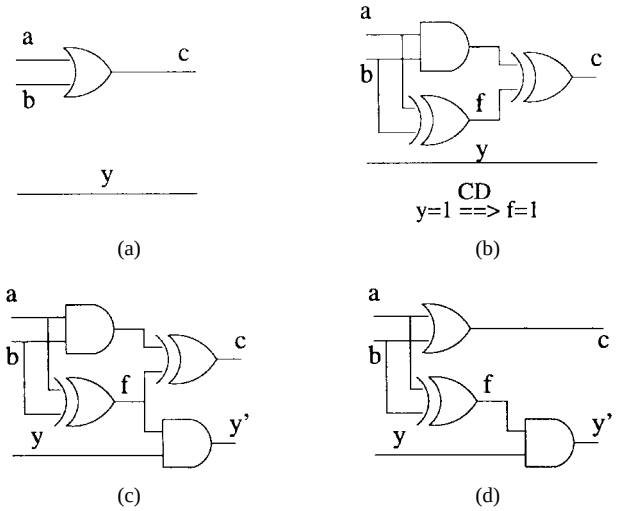

Fig. 5. Example of an IB transformation using Reed–Muller expansions.

can combine lines through an additional EX-OR gate like it is done in Transformations 1–4. EX-OR gates can be introduced as permissible functions based on conditions testing for line y s–a-0 as well as the line y s-a-1. The following transformations give the condition for introducing permissible EX-OR gates.

Theorem 4.3: Let f and y be arbitrary nodes in a combinational network, C, where f is not in the transitive fanout of y, and y is an irredundant node in the circuit. The function y' with $y' = y|_1 \oplus f$ is a permissible function at node y if and only if the CD relations $y = 0 \overset{\text{CD}}{\rightarrow} f = 0$ and $y = 1 \overset{\text{CD}}{\rightarrow} f = 0$ are true. □

Theorem 4.4: Let f and y be arbitrary nodes in a combinational network, C, where f is not in the transitive fanout of y, and y is an irredundant node in the circuit. The function y' with $y' = y|_1 \oplus \overline{f}$ is a permissible function at node y if and only if the CD relations $y = 0 \overset{\text{CD}}{\rightarrow} f = 1$ and $y = 1 \overset{\text{CD}}{\rightarrow} f = 1$ are true. □

Transformations 7 and 8: Transformations based on Theorems 4.3 and 4.4, followed by redundancy removal.

Example 4.2: The circuit shown in Fig. 4 illustrates an example for Transformation 8. The conditions required to test d s-a-0, as well as d s-a-1, both include the assignment $(a = 1)$. Thus, from the definition of CD relations, we get $d = 0 \overset{\text{CD}}{\rightarrow} a = 1$ and $d = 1 \overset{\text{CD}}{\rightarrow} a = 1$ as true. The resultant circuit is shown in Fig. 4(b). □

The transformations introduced here are new and can be used for area optimization as well as for random testability enhancement in combinational circuits. Introduction of the EX-OR gate may result in more than one line becoming redundant, thus aiding area optimization. The EX-OR gate will also enhance the observability of all the faults in the input cone of f, resulting in better random testability.

C. Transformation Based on Reed–Muller Expansions

Transforming AND and OR gates into Reed–Muller expansions can lead to different circuit representations from where the intermediate nodes can be used as "divisors." Two-input AND gates and OR gates can be expanded by the relations

$$\mathrm{A} + \mathrm{B} = \mathrm{A} \oplus \mathrm{B} \oplus (\mathrm{A}.\mathrm{B})$$
$$\mathrm{A}.\mathrm{B} = \mathrm{A} \oplus \mathrm{B} \oplus (\mathrm{A} + \mathrm{B}).$$

To represent the expanded relations, the output of the new gates introduced can become part of an IB transformation. These transformations are different from those used in [4] and hence can aid in circuit optimization and random testability enhancement.

Transformation 9: Expand node functions using Reed–Muller expansions, use intermediate nodes for IB transformations, and then reduce the expanded function back to original form.

Given below is an example to yield better understanding to these types of transformations.

Example 4.3: Consider a subcircuit, shown in Fig. 5(a). Let the OR gate in the subcircuit be expanded, based on its Reed–Muller expansion, as shown in Fig. 5(b). Now, it may happen that in the subcircuit, there is a CD relation, denoted by $y = 1 \overset{\text{CD}}{\rightarrow} f = 1$. The circuit transformation based on the CD relation is shown in Fig. 5(c). The final subcircuit, after extracting back the OR gate, is in Fig. 5(d). The transformation from the subcircuit in Fig. 5(a) to the one in Fig. 5(d) is new and may lead to a different search space in circuit optimization. Moreover, stuck-at faults at a and b are observable through the AND gate whenever y is one. This would enhance the detection probabilities of all the faults in the fanin cone of a and b. □

Reed–Muller expansions can be further extended to multi-input gates, which can allow a large number of intermediate functions.

V. Synthesis for Area Optimization

Here, the main emphasis during optimization is on area only. The new transformations introduced in the paper are used in combination with the existing IB transformations

LOT - Area Optimization($Iter$)
Begin
1. For $i = 1$ to $Iter$ do
2. For each line y in the circuit do
3. Search for next possible indirect implication;
4. Make the appropriate transformation 1-8;
5. If area reduced or same, accept, else repeat;

End

Fig. 6. Algorithm LOT for area optimization.

[4] to obtain a stronger combinational multilevel area optimization tool. Application of EX-OR-based transformations is motivated by the fact that optimized AND-EXOR expressions require fewer products than sum-of-product expressions in many circuits [29]. Efficient library implementation of EX-OR gates can ensure that their extra delay and area overhead is not significant compared to other gates. EX-OR gates can be particularly useful if they can replace several two-input gates. For the implementation for area optimization, we have restricted ourselves to Transformations 1–8. As the Reed–Muller based transformations were found computationally intensive, they were applied to selective areas to improve testability as shown in the next section. Transformations made on the circuit are accepted or rejected based on the resultant area of the circuit. This method is greedy in nature and accepts any of the transformations which can reduce the area. The new transformations based on Transformations 5–8 can cover new search spaces during optimization and, thus, have the potential to yield a smaller circuit. The procedure stops after a predetermined number of iterations across the circuit.

TABLE II
COMPARISON OF AREA WITH SIS 1.2 AND HANNIBAL

Circuit Name	Original Area	SIS 1.2 Area	HANN. Area	LOT Area	Time hr:min	Comparison LOT/SIS	Comparison LOT/HAN
c432	4152	4048	3352	3424	00:05	0.845	1.021
c499	6976	11049	6696	7360	00:01	0.666	1.099
c880	8976	9160	8296	7416	00:03	0.809	0.893
c1355	12800	11264	12456	7360	00:03	0.590	0.653
c1908	15272	10440	10520	8840	00:15	0.846	0.840
c2670	24920	16936	17048	15680	00:20	0.925	0.919
c3540	33824	30008	26632	25208	02:20	0.840	0.948
c5315	55024	39480	44224	41952	01:39	1.062	0.948
c6288	59520	66072	58200	57808	06:10	0.874	0.993
c7552	73056	50160	41680	38192	10:11	0.761	0.916

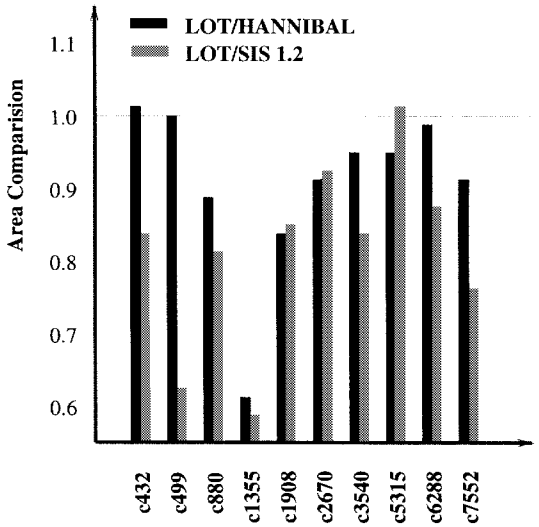

Fig. 7. Area savings compared to SIS 1.2 and HANNIBAL.

A. Experimental Results

Experiments were conducted on the ISCAS 85 benchmark circuits to demonstrate the combined strength of the transformations. Our tool, LOT, has been implemented by making extensions to the HANNIBAL code [4], which also includes the public domain fault simulator FSIM [5]. A short pseudocode of the implementation has been presented in Fig. 6. For cost estimation during synthesis, technology mapping would have given the best estimate of the circuit area—but it would not have been efficient. Instead, we have used the number of two-input gates, denoted as 2-i/p GE (equivalent to half the number of connections in [4]) to evaluate the cost. A two-input EX-OR gate has been counted as 2.5 two-input gates as in [36]. After the synthesis procedure is complete, however, all the circuits were mapped to a library to compare the areas of the circuits after technology mapping. The library used is derived, from *stdcell2_2.genlib*—the library and its parameters are described in the Appendix. The results, presented in Table II, show that the new EX-OR-based transformations can make an important difference in the area of the circuits, particularly when EX-OR gates can be accommodated by the library. The circuits synthesized by the proposed tool, LOT, were compared with those synthesized by SIS 1.2 (*script.rugged*) and HANNIBAL [4]. It may be seen that there is a clear potential for smaller area by LOT when compared to SIS 1.2 or HANNIBAL. It is interesting to note that the amount of savings for LOT over SIS 1.2 has a similar pattern to that over HANNIBAL, as shown in Fig. 7. On the average, LOT reduced the area over HANNIBAL by 8% and over SIS 1.2 by 18%. The synthesis time, as reported in the sixth column, was measured in a Sparc 5 machine and is comparable to [4]. This run-time can be significantly improved by a better software implementation techniques. Also, one of the areas we can obtain major savings in time is in redundancy removal as seen in [10]. One other tradeoff that needs to be studied is increasing the maximum level of recursion learning which can yield more indirect implications and therefore, more flexibility in the search for optimality.

Though the synthesis tool does not use path delay in the cost function, the "longest sensitizable" path lengths of the resultant circuits were measured [37] and presented in Table III. The path length is counted in terms of the number of levels in the unmapped circuit. It can be observed that the length of the critical paths do not vary drastically and appears to be within acceptable limits. Also, the synthesis process using IB transformations can be modified to ensure that a given list of critical paths remain unchanged during synthesis.

Table IV presents the number of EX-OR gates that were used in the optimized circuits. The number of two-input EX-OR gates has been presented in Column 3. Column 4 gives the total

TABLE III
ESTIMATE OF THE LONGEST PATH LENGTHS

Circuit Name	Longest Sensitizable Path Length		
	Original	HANNIBAL	LOT
c432	19	23	19
c499	17	19	21
c880	24	29	35
c1355	24	24	21
c1908	37	31	27
c2670	30	22	24
c3540	45	41	39
c5315	47	41	51
c7552	40	86	60

TABLE IV
ESTIMATE OF THE NUMBER OF EX-OR GATES

Circuit Name	2-i/p EX-OR Gates		Total # of 2-i/p Gates
	Original	LOT	
c432	18	0	105
c499	104	104	172
c880	0	27	245
c1355	0	104	172
c1908	0	63	234
c2670	0	52	442
c3540	0	131	780
c6288	0	15	1841
c7552	0	283	943

number of two-input equivalent gates in the circuit (as has been measured in [9]). Here, a two-input EX-OR gate is counted as a two-input gate. As can be observed, LOT incorporates many EX-OR gates in the optimized circuit and uses them in the circuit. As an example, it could identify all the EX-OR gate structures in c1355, and the resultant structure is now similar to c499 (c1355 was derived from c499, by transforming its EX-OR gates to NAND gate representations).

VI. RANDOM PATTERN TESTABILITY ENHANCEMENT

The following describes the integrated framework for the tool to simultaneously optimize area and enhance testability. The underlying philosophy here is to integrate the primary objective of optimal area with a secondary objective of testability. The proposed framework, however, allows for reordering and expanding the objectives. For example, if the primary objective is testability and the secondary objective is area, the ordering of the optimization procedures can be changed, as shown here.

Effective BIST requires design of random pattern testable circuits or design of test pattern generators (TPG's), tailored for the circuit under test (CUT). Those faults that need long TPG sequences for testing are termed hard-to-detect (HTD) faults. A less randomly testable circuit will have a high number of HTD faults. Standard techniques used to test HTD faults, using shorter TPG sequences, use complex additional hardware, either inside the circuit with additional pins (test points) or in the TPG [32]. The additional hardware incurred can be minimized if the circuits are synthesized with a higher random pattern testability.

During synthesis, most of the transformations applied here would result in reducing the area, but some may increase the area to sacrifice for better testability. The two main features of our procedure include a synthesis guidance procedure based on random pattern testability and a method for efficient detection of the cost function of the circuit. The cost function here will depend on the fault detection probabilities in the circuit, and an approximate estimate is used to guide the search.

A. Calculating Detection Probabilities

As the proposed synthesis procedure is guided by the random testability of the combinational circuit at each iteration, a fast estimation of the detection probabilities of the faults is very important. Though exact calculation of fault detection probabilities is done by signal probability calculations [22], [32], an approximate and quick estimate is more practical for our synthesis methodology. This method is based on statistical estimation of fault detection probabilities, as proposed in STAFAN [26]. STAFAN uses results on controllability of lines, as well as statistics on sensitization frequency of lines, computed by simulating a given set of patterns against the logic, to deduce fault detection probabilities. A s-a-0 (s-a-1) detection probability is estimated as the product of 1-controllability (0-controllability) and 1-observability (0-observability) of the line. The 1-observability (0-observability) of a line y is the probability of observing a 1(0) on line y at a primary output. The 1-controllability (0-controllability) of a line y, denoted by $CO_1(y)$ $[CO_0(y)]$, is estimated statistically. The 1-observability (0-observability) of a line y, denoted by $OB_1(y)$ $[OB_0(y)]$, is estimated from the one-level path sensitization probability $U(y)$ of the line and other controllability parameters. The one-level path sensitization probability $U(y)$ of the line y is the probability that a given input vector will sensitize the value on the line to the output of the gate to which it is connected. The U value of an output line, or a fanout stem, is defined as one. Simulations are done for the circuit to estimate the controllabilities and the U values of each line. The method of calculating the observabilities from the estimated parameters is described in [22] and [26]. STAFAN was chosen as it can provide fast and efficient test analysis of the circuit at each iteration. There can be cases, however, where the predicted test analysis may differ from the actual values [33]. These differences are typically caused by redundancies which would not occur in the synthesized circuits. One can also choose any alternative test analysis tool with this methodology.

The proposed method begins by applying STAFAN to estimate the fault detection probabilities and to identify the HTD faults in the given multilevel circuit. Whenever the circuit is changed structurally to a functionally equivalent circuit, there will be a change in the estimated parameters in the circuit. If the testability has to be measured at each iteration of the synthesis process, estimating the parameters every time will be time consuming. The proposed synthesis method uses some incremental updates on earlier estimated parameters based on the nature of transformation made on the circuit. Calculation of fault detection probabilities based on the estimated parameters is done in a single traversal through the graph [26]. Presented below are techniques to update the

Update_Stafan(Transformation Type) /* Redundancy Elimination Stage */
Begin
1. Set the CO_1 (CO_0) of all redundant s-a-1 (s-a-0) lines to 1.0;
2. For each node l of the circuit in the topologically sorted order do
3. Update CO values of l based on the nature of the gate from which it is connected;
4. Update U values of the inputs to the gate at l;

End

Fig. 8. Algorithm to estimate CO values and U values during redundancy elimination.

CO values and the U value of each node in the circuit, based on the IB transformations.

As the IB transformation is a two-step process, transformation and reduction, updates on the estimated parameters are also done in two steps. For the first step, transformations are based on Theorem 3.1. The updates made in a line y, during redundancy elimination, are related to the change of the parameters in the fanin cone of the line y. The update procedure is described in the algorithm **Update_Stafan** (Fig. 8).

The procedure first sets the $CO_1(CO_1)$ values of the s-a-1 (s-a-0) redundant lines equal to one. This would create a change in the CO values of the redundant lines, compared to its values before the transformation. Any change in the CO values at line y will cause a change in the CO values at the outputs of the gate to which it is connected. Moreover, the change in the CO values will also cause a change in the U values of all the inputs to the gate to which it is connected. The nature of the change in these parameters depends on the nature of the gate line y to which it is connected. As an example, consider a k input AND gate. The updates related to an OR gate, NOR gate, or a NAND gate can be determined in a similar manner. Denote the inputs to the AND gate by $a_1, a_2, \cdots, a_k$, and output by b. Assume a change in the controllability of line a_u and determine its effect on the CO values of b and the U values of $a_1, a_2, \cdots, a_k$. If no fanout exists in the circuit, $CO_1(b)$ can be estimated by the following equation:

$$CO_1(b) = \prod_{i=1}^{k} CO_1(a_i). \tag{2}$$

Moreover, the U value at a_j is estimated by

$$U(a_j) = \prod_{i \neq j}^{i=0 \text{ to } k} CO_1(a_i). \tag{3}$$

The proposed method calculates the estimate updates, even in the presence of fanouts in the circuit and is hence, approximate. A change in $CO_1(a_i)$ by an amount $\Delta CO_1(a_i)$ would result in the following changes for an AND gate in the CO values and U values:

$$\begin{aligned} \Delta CO_1(b) &= [\Delta CO_1(a_u)] \times \left[\prod_{i \neq u}^{i=0 \text{ to } k} CO_1(a_i) \right] \\ \Delta U(a_u) &= 0 \\ \Delta U(a_j) &= [\Delta CO_1(a_u)] \times \left[\prod_{i \neq u, j}^{i=0 \text{ to } k} CO_1(a_i) \right] \\ & \qquad \forall j \quad \text{and} \quad j \neq u. \end{aligned} \tag{4}$$

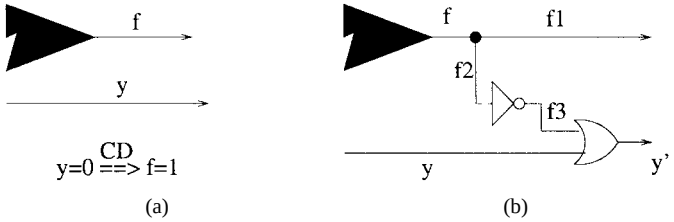

Fig. 9. Transformation 1—An example.

When more than one input to the gate has changed its controllability, the update procedure will take one input at a time. The updates should take into account that $CO_1(b)$ has to be less than or equal to the CO_1 values of all its inputs. The updated U values at the gate inputs should always be greater than or equal to $CO_1(b)$ [26]. The update procedure involving each gate is linear with the fanin of the gate.

The update procedure during transformations depends upon the nature of the "implication" relationship on which it is based. The type of the transformation determines the way the values are updated. Given below are the steps incurred by the update procedure for the Transformation 1. The steps for update procedures based on Transformations 2–4 will be similar.

Consider the subcircuit shown in Fig. 9(a). A transformation based on the CD relation $y = 0 \overset{\text{CD}}{\rightarrow} f = 1$ (Transformation 1) would result in the subcircuit shown in Fig. 9(b). As can be observed, there is a NOT gate and an OR gate added in the new subcircuit, resulting in four new lines, namely, f_1, f_2, f_3, and y'. The updated CO values and the U values are expressed as

$$\begin{aligned} CO_1(f_1) &= CO_1(f_2) = CO_0(f_3) = CO_1(f) \\ CO_0(f_1) &= CO_0(f_2) = CO_1(f_3) = CO_0(f) \\ CO_0(y') &= CO_0(y);\ CO_1(y') = CO_1(y) \\ U(f_1) &= U(f);\ U(f_3) = CO_1(y) \\ U(y') &= U(y);\ U(y) = CO_1(f). \end{aligned}$$

Because these estimates are determined regardless of whether the transformation was based on an equivalent or permissible function, the estimates are approximate. The proposed method uses simulation for better parameter estimation, after every predetermined number of incremental updates on the parameters.

Testability Cost: The testability cost (TC) takes into account the change in the fault detection probabilities of the HTD faults, as these faults impact on the random pattern test length. In the proposed method, HTD faults are those

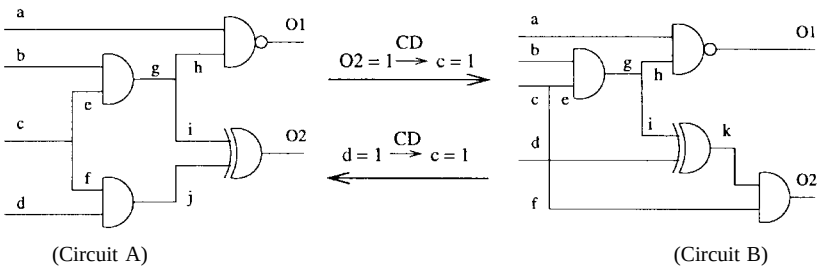

Fig. 10. IB transformations for Example 6.1.

faults whose fault detection probabilities lie between $PD\_\min$ and ($PD\_\min \times PD\_range$). $PD\_\min$ is equal to the minimum fault detection probability. PD_range is given as an input parameter to the synthesis process. Thus, the TC of a transformation is given by

$$\text{TC} = \sum_{\forall l \in HTD} \text{NEW\_PD}(l) - \text{OLD\_PD}(l). \qquad (5)$$

B. Testability Enhancement Algorithm

Proposed is an algorithm that can be used to optimize area with enhancement of random pattern testability. IB transformations which enhance random testability can be classified into two types.

1) *Type A:* IB transformation enhancing testability which requires *no* increase in area.
2) *Type B:* IB transformation enhancing testability which may require increase in area.

The proposed method uses both these transformations iteratively in the synthesis process. It has been observed that as a circuit is transformed during logic synthesis to improve area or speed, its testing property can change, often in unpredictable ways [23]. The most important factor in such a procedure is the ability to identify transformations that are good for the random testability of the circuit.

Example 6.1: Consider two circuits, A and B, in Fig. 10. Circuit A can be transformed to Circuit B using the transformation based on (line $O1 = 1$) $\stackrel{\text{CD}}{\rightarrow}$ (line c = 1). Furthermore, circuit B can be transformed back to circuit A, using the transformation (line d = 1) $\stackrel{\text{CD}}{\rightarrow}$ (line c = 1). It can be observed that both the circuits have the same area in terms of connections and literal counts. Here, one can verify whether one of the circuits is better than the other in random pattern testability. The detectability profile of the circuits is presented in Fig. 11. Note that circuit A has two HTD faults (faults with only two test vectors), compared to circuit B having three HTD faults. If we assume that all the test vectors are equiprobable, circuit A is a more randomly testable circuit compared to circuit B. Any synthesis method optimizing area or delay may choose circuit B instead of circuit A, and degrade random pattern testability of the circuit. If the circuit is a subcircuit of a larger circuit, the test vectors at the input of the subcircuit may not be equiprobable and the random testability might be different.

Type A Transformations: IB transformations of Type A are the area-optimization Transformations 1–8. In the proposed method, among these, only those transformations are accepted where the estimated testability cost (5) is positive. Thus, area optimization is allowed with improvement of random pattern testability of the circuit.

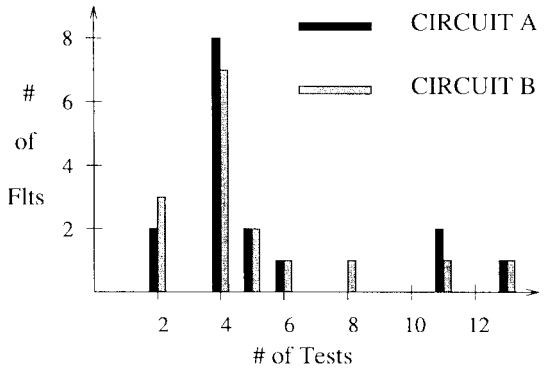

Fig. 11. Detectability profile. (a) Circuit A. (b) Circuit B.

Type B Transformations: IB transformations of Type B, where the area can be increased to enhance testability, involve addition of redundant lines which improve the observability of areas in the circuit containing a high concentration of HTD faults. These transformations involve the addition of some redundant lines to improve the observability of hard-to-detect faults. Though these transformations may introduce redundant lines in the circuit, the synthesis procedure ensures that these lines are not removed in the next few iterations during redundancy elimination. Subsequent transformations make most of these lines irredundant in the circuit. Observability is enhanced using two types of transformations:

Observability Enhancement Based on Transformations 1–4: Extra connections can be introduced based on Transformations 1–4 to enhance observability. Fig. 9 gives the example of Transformation 1. Note that such a transformation enhances the observability at the vicinity of the fanin cone of f: the shaded area in the figure. If this area consists of considerably fewer HTD faults, the transformation does not make any improvement in the random testability of the circuit. Transformations where the area at the vicinity of the cone of f has a large concentration of HTD faults and where y has a high observability are useful in enhancing random pattern testability. The procedure tries to introduce a large number of such redundant lines and ensures that these lines are not removed in the next few iterations during redundancy elimination.

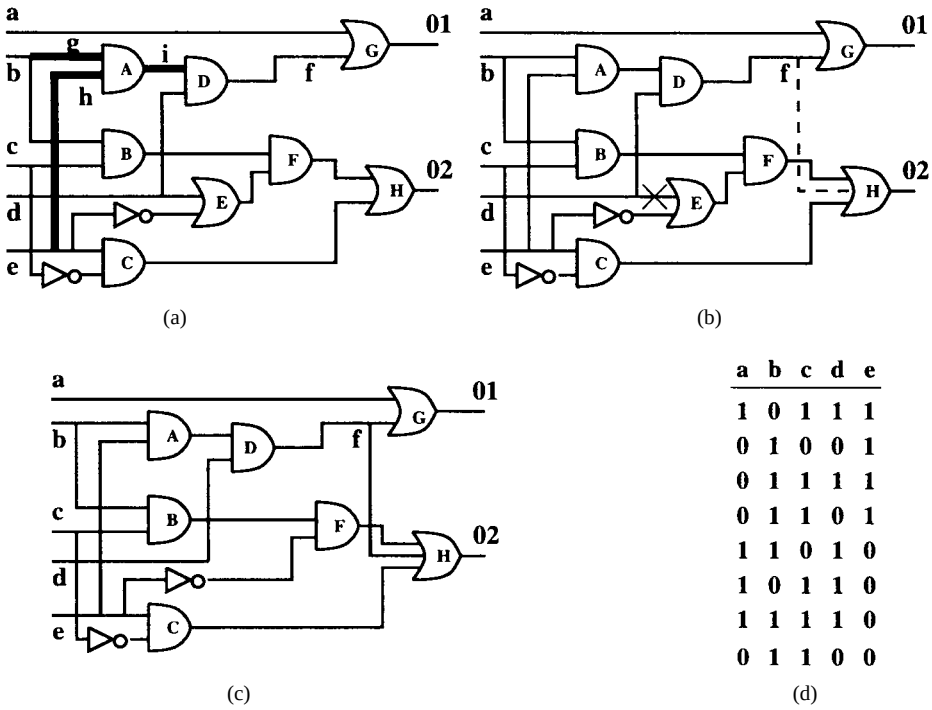

Fig. 12. Example illustrating an observability enhancing transformation. (a) Original circuit, (b) circuit after Transformation2, (c) final circuit, and (d) test vectors.

Example 6.2: The circuit in Fig. 12(a) has six faults having a detection probability of less than 0.1, including three in the fanin cone of gate D. The three HTD faults, g s-a-1, h s-a-1, and i s-a-1 cannot be tested in the given test set [Fig. 12(d)], as an instance. The untestable fault lines are denoted by bold lines in the figure. A transformation possible here based on the relation $O2 = 0 \overset{\text{CD}}{\rightarrow} f = 0$ can improve the observability of the fanin cone of D. The connection as shown in Fig. 12(b) also results in a redundant line at the input of gate E. The final circuit in Fig. 12(c) has only three faults of detection probability less than 0.1. In fact, all the faults can now be tested in the given test set. Note that the number of two-input gates in the initial circuit and the final circuit is the same and there is now a difference in the testability. □

Observability Enhancement Based on Reed–Muller Expansions: Using Transformation 9 will enable creation of different intermediate circuit structures and connections that can improve the observability of HTD fault zones. Moreover, these transformations introduce more EX-OR gates in the circuit.

Example 6.3: Consider a circuit shown in Fig. 13(a). This circuit has three faults with a detection probability less than 0.1. Given an instance of a test set, as in Fig. 13(d), among the three, two of the faults, h s-a-1 and i s-a-1 cannot be tested (denoted by bold lines). To improve the testability, a Reed–Muller based transformation can be used as in Fig. 13(b). Here, such a transformation was possible as $O1 = 0 \overset{\text{CD}}{\rightarrow} (e \oplus f) = 0$. The new connection, denoted by the broken line, improves the observability at the fanin cone of gate C. The connection also results in a redundant line at the input of gate A. The final circuit is given in Fig. 13(c). Thus a new EX-OR gate has been introduced, the EX-OR gate improves the testability and it was found that all the stuck-at faults can be tested by the *test set* in Fig. 13(d). □

C. Synthesis Algorithm

The synthesis procedure, as shown in Fig. 14, has two phases. In Phase 1, area optimization is done on the circuit, based on the earlier proposed IB transformations (1–8). In Phase 2, we use Type B and Type A transformations to realize testability enhancement. The synthesis procedure starts with Phase 1 and continues reducing the circuit size until further reduction would reduce its testability. At this point, the procedure changes to Phase 2 (Steps 12 and 13) and continues with transformations with testability enhancement (Steps 8–11). Phase 1 minimizes the area without degrading the random testability, whereas Phase 2 tries to improve the random testability. Finally, any line introduced by Type B transformations and left redundant will be removed in Step 14.

Though the above procedure concentrates only on the testability, the emphasis on area optimization can also be easily controlled. Our synthesis method can iteratively use Phases 1 and 2 until the design goals are met. Moreover, in this synthesis methodology we have not considered the delay of the resulting circuit. The IB transformations presented here can also be used under the added constraint of the delay of the network.

D. Experimental Results

Experiments were performed on PLA benchmarks as well on some ISCAS benchmark circuits. Those circuits were chosen which had poor testability and contained a number of random-pattern-resistant faults. The synthesis algorithm begin

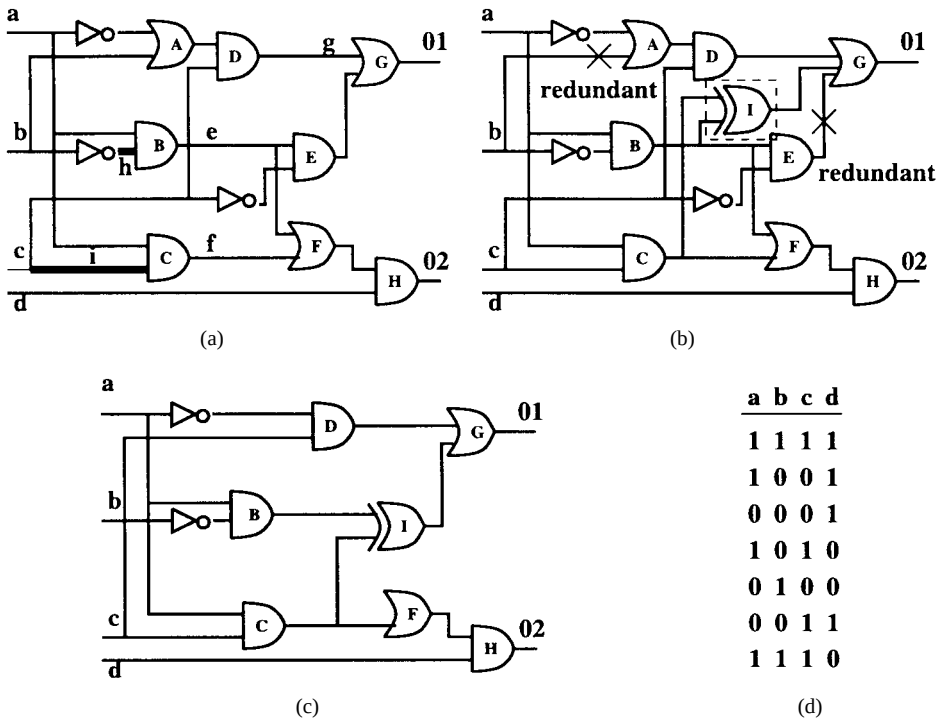

Fig. 13. Example illustrating observability enhancement. (a) Original circuit, (b) circuit after Transformation 9, (c) final circuit, and (d) test vectors.

LOT - Random Testability(*Iter*)
Begin
1. Apply *Stafan()* on the initial circuit;
2. Phase_Status = Phase 1;
3. For $i = 1$ to *Iter* do
4. If (Phase_Status = Phase 1) Then
5. For each line y in the circuit do
6. Make Transformations 1 - 8;
7. Else
8. For each line y in the circuit do
9. Make *Type B* Transformations /* Uses Transformation 9 */;
10. For each line y in the circuit do
11. Make *Type A* Transformations;
12. If (Testability reduced in last loop) then
13. Phase_Status = Phase 2;
14. Remove any additional redundant line;

End

Fig. 14. Algorithm LOT for enhancing testability.

with multilevel circuits and improve the testability based on a given criteria. For the PLA benchmarks, the tools used circuits, synthesized by *tstfx* [21], as their input, and optimized them.

To make the circuits compatible to our implementation, the multilevel circuits synthesized by *tstfx* were mapped to a SIS library [16] obtained from *mcnc.genlib* which had no complex gates. The mapped circuits were then synthesized by LOT. For the sake of comparison, these circuits were also synthesized by HANNIBAL [4], SIS 1.2 [16] (using *script.boolean and script.rugged*) which only optimizes area. The test lengths of the circuits were obtained by using LFSR-generated pseudorandom sequences. The test sequences for 100% single stuck-at fault coverage presented are an average,

TABLE V
100% FAULT COVERAGE TEST LENGTH AND AREA COMPARISON WITH *tstfx* AND SIS 1.2

Ckt.	*tstfx*		SIS 1.2		LOT		*tstfx*/LOT		SIS/LOT	
	Ar	TL	Ar	TL	Ar	TL	Ar	TL	Ar	TL
b10	382	19.0K	346	17.8K	280	17.8K	1.36	1.06	1.23	1.00
b4	250	34.9K	230	56.4K	189	34.9K	1.32	1.00	1.22	1.62
gary	424	15.5K	410	18.0K*	288	14.3K	1.47	1.08	1.43	1.25
in2	368	12.5K	356	12.5K*	201	4.8K	1.83	2.60	1.77	2.60
in4	440	689K	356	63.7K	197	63.7K	2.23	10.8	1.81	1.00
in5	227	57.8K	221	14.4K*	169	12.8K	1.34	4.51	1.30	1.13
in6	256	34.9K	239	56.4K	190	34.9K	1.37	1.00	1.25	1.62
in7	100	116K	88	116K	59	34.7K	1.69	3.35	1.49	3.35
vg2	177	1657K	100	126K	61	126K	2.90	13.1	1.64	1.00
x1dn	251	1191K	85	340K	90	189K	2:78	6.30	0.94	1.79
Average							1.83	4.48	1.41	1.66

* denotes that circuits have redundant faults

over several different LFSR-generated sequences (typically, five to ten primitive polynomials were used). For selected circuits including some ISCAS benchmark [25] circuits, which had a test length more than half a million vectors, testability is measured by the number of undetected (testable) faults. The circuit areas (Ar) were compared, based on two-input GE's. The results are presented in Tables V–VII.

Comparison with tstfx: As can be observed from Table V, the proposed method improves both the area and the test length of the circuits synthesized by *tstfx*. The area of the *tstfx*-synthesized circuits was further reduced by 83%, on the average, by LOT. For seven out of the ten circuits, there was a considerable reduction in the test length by the synthesis tool—the average ratio being equal to 4.48 : 1. In particular, circuits with a high test length (in4, in7, vg2, x1dn) have shown considerable improvement in testability.

Comparison with SIS 1.2: The ISCAS circuits and the circuits from the PLA benchmark circuits were optimized by

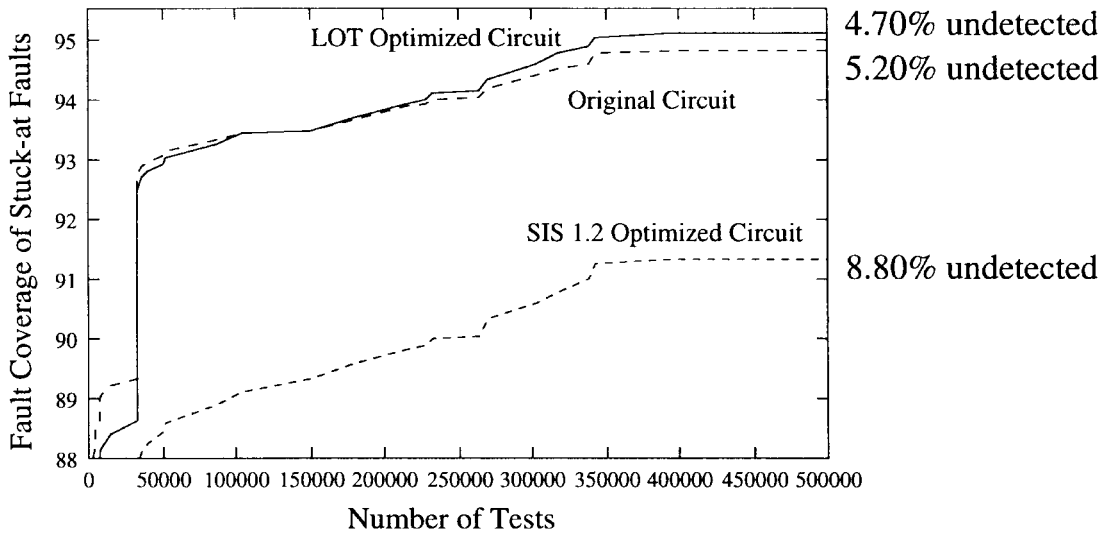

Fig. 15. Fault coverage for c2670 with optimization.

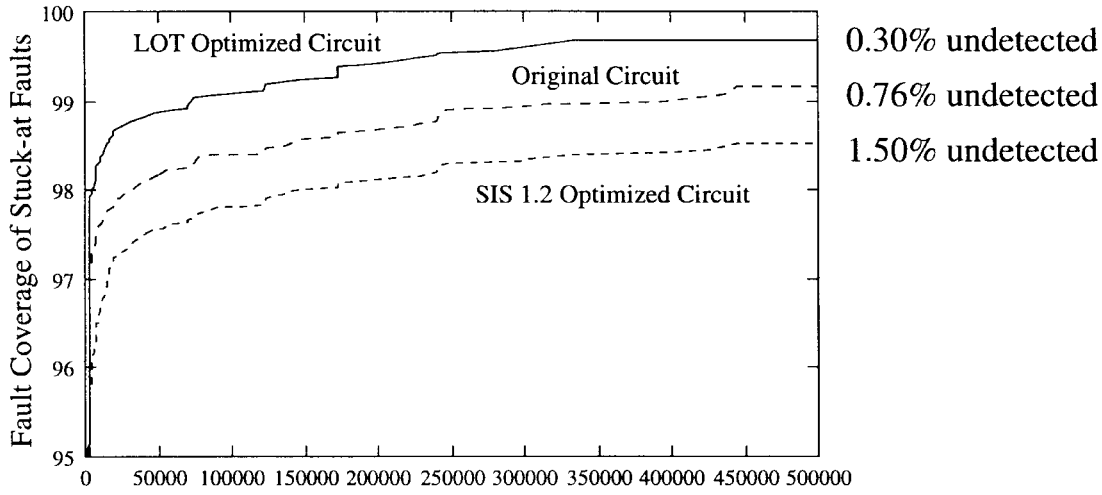

Fig. 16. Fault coverage for c7552 with optimization.

TABLE VI
TESTABILITY COMPARISON OF RANDOM-PATTERN RESISTANT ISCAS CIRCUITS

Circuit	Original		SIS 1.2		LOT	
	Ar	TL/Undet.	Ar	TL/Undet.	Ar	TL/Undet.
s641	133	224K/2	142	224K/1	129	224K/1
s713	169	224K/40	142	224K/1	129	224K/1
s1196	434	180K/0	475	180K/0*	367	180K/0
rckl	188	482K/205	139	404K/125	125	482K/98
c2670	779	499K/110*	567	499K/156*	520	499K/75
c3540	1113	22K/0*	1069	27K/0*	835	22K/0
c7552	2367	452K/48*	1822	444K/73*	1367	444K/8

* denotes that circuits have redundant faults

TABLE VII
100% FAULT COVERAGE TEST LENGTH AND AREA COMPARISON WITH HANNIBAL

Circuit	HANNIBAL		LOT		HANNIBAL/LOT	
	Ar	TL	Ar	TL	Ar	TL
b10	280	17.8K	280	17.8K	1.00	1.00
b4	179	56.4K	189	34.9K	0.95	1.62
gary	288	18.0K	288	14.3K	1.00	1.25
in2	180	11.7K	201	4.8K	0.89	2.43
in4	197	63.7K	197	63.7K	1.00	1.00
in5	159	57.8K	169	12.8K	0.94	4.51
in6	180	56.4K	190	34.9K	0.95	1.62
in7	59	88.2K	59	34.7K	1.00	2.54
vg2	61	126K	61	126K	1.00	1.00
x1dn	69	340K	90	189K	0.77	1.79
Average					0.95	1.88

SIS 1.2. Both the scripts, script.boolean and script.rugged, were used and the better implementation is reported here. The comparison of the PLA benchmark circuits in Table V show that LOT optimized circuits have 41% lesser area and 66% smaller test length. For the ISCAS circuits, in Table VI, LOT gives a much lesser area and lesser undetected faults. For seven out of the 17 circuits considered, it was found that SIS 1.2 degrades the testability of the circuit during optimization. Also seven circuits synthesized by SIS 1.2 were found to contain redundant faults after optimization.

TABLE VIII
DESCRIPTION OF THE LIBRARY USED FOR TECHNOLOGY MAPPING

Type	Name	Size	Function	Other parameters
GATE	invf101	16	O=!A1	INV 1 999 1 .2 1 .2
GATE	norf201	24	O=!(A1+B1)	INV 1 999 1 .2 1 .2
GATE	norf301	32	O=!(A1+B1+C1)	INV 1 999 1 .2 1 .2
GATE	nanf201	24	O=!(A1*B1)	INV 1 999 1 .2 1 .2
GATE	nanf301	32	O=!(A1*B1*C1)	INV 1 999 1 .2 1 .2
GATE	nanf211	32	O2=A1*B1	NONINV 1 999 1 .2 1 .2
GATE	nanf311	40	O2=A1*B1*C1	NONINV 1 999 1 .2 1 .2
GATE	norf211	32	O1=A1+B1	NONINV 1 999 1 .2 1 .2
GATE	norf311	40	O1=A1+B1+C1	NONINV 1 999 1 .2 1 .2
GATE	xorf201	48	O=!(A1*B1+!A1*!B1)	UNKNOWN 1 999 1 .2 1 .2

Comparison with HANNIBAL: HANNIBAL optimizes circuits on the basis of area. When the same circuits are optimized for testability as in LOT, the area of the resultant circuits were slightly higher compared to HANNIBAL, as shown in Table VII. The increase in area is not significant, however, with the average being equal to 5%. On the contrary, the test length for 100% fault coverage was reduced by 88%. Though, in many circuits, optimization by HANNIBAL can reduce the test length, the test length was found to increase in three out of the ten circuits.

Therefore, in summary, LOT can achieve significantly better testability with area compared to available synthesis tools like *tstfx* and SIS 1.2. In fact, compared to HANNIBAL, which only synthesizes for area, LOT can synthesize more testable circuits with little penalty in area. Fault coverages for the two random-pattern-resistant ISCAS85 [25] circuits, c2670 and c7552 are presented in Figs. 15 and 16. Most of the HTD faults that could not be eliminated can be denoted as functionally HTD faults, i.e., they are a property of the circuit function and any implementation of the circuit will have those faults (or equivalent faults). To illustrate a case, the HTD faults at the primary inputs and primary outputs are *functionally HTD faults*. Some of the AND gates or OR gates with large fanins, near the primary inputs, may result in HTD faults of this nature. For example, c2670 has four AND gates with fanin 15, which results in most of the HTD faults after optimization using LOT. During optimization, however, if new HTD faults are created as in SIS 1.2, the testability of the circuit worsen.

VII. CONCLUSIONS

A novel optimization procedure using recursive learning-based implications is presented here. The transformations proposed are based on a) new logic transformations based on EX-OR gates, b) methods to guide the synthesis process enhancing random pattern testability, and c) heuristics to identify difficult-to-test faults. The new EX-OR logic transformations, which were developed with the primary objective of enhancing observability difficulties, also play an important role in area optimization. Our synthesis scheme can be applied to poor random-testable multilevel circuits, which were synthesized by another synthesis tool, to improve random testability, as well as area. Experiments indicate improvement over other optimization tools, in terms of area as well as testability.

APPENDIX
LIBRARY FOR TECHNOLOGY MAPPING

The library used for technology mapping has been derived from *stdcell2_2.genlib* [16]. Table VIII, presents the contents of the library elements. The library only consists of simple logic gates. The only complex gate used here is a two-input EX-OR gate.

REFERENCES

[1] W. Kunz and D. K. Pradhan, "Recursive learning: A new implication technique for efficient solution to CAD problems—Test, verification and optimization," *IEEE Trans. Computer-Aided Design,* vol. 13, no. 9, pp. 1143–1158, Sept. 1994.
[2] D. K. Pradhan and W. Kunz, "Method for circuit verification and multi-level circuit optimization based on structural implications," US Patent 05/526 514, June 11, 1996.
[3] W. Kunz, D. K. Pradhan, and S. Reddy, "A novel framework for logic verification in a synthesis environment," *IEEE Trans. Computer-Aided Design,* vol. 15, no. 1, pp. 1–15, Jan. 1996.
[4] W. Kunz and P. R. Menon, "Multilevel logic optimization by implication analysis," in *Int. Conf. Computer-Aided Design,* San Jose, CA, 1994, pp. 6–13.
[5] H. K. Lee and D. S. Ha, "A efficient forward fault simulation algorithm based on the parallel pattern single fault propagation," in *Proc. Intl. Test Conf.,* Washington, DC, 1991, pp. 946–953.
[6] V. K. Agarwal and E. Cerny, "Store and generate built-in testing approach," in *Proc. Fault Tolerant Computer Symp.,* Boston, MA, June 1981, pp. 35–40.
[7] S. Pateras and J. Rajski, "Cube-contained random patterns and their application to the complete testing of synthesized multi-level circuits," in *Proc. Int. Test Conf.,* Washington, DC, 1991, pp. 473–481.
[8] L. A. Entrena and K. T. Cheng, "Combinational and sequential logic optimization by redundancy addition and removal," *IEEE Trans. Computer-Aided Design,* vol. 14, no. 7, pp. 909–916, July 1995.
[9] S. C. Chang and M. Marek-Sadowska, "Perturb and simplify: Multi-level boolean network optimizer," in *Int. Conf. Computer-Aided Design,* San Jose, CA, 1994, pp. 2–5.
[10] S. C. Chang, L. P. P. P. Van Gineeken, and M. Marek-Sadowska, "Fast boolean optimization by rewiring," in *Int. Conf. Computer-Aided Design,* San Jose, CA, 1996, pp. 262–269.
[11] S. Muroga, "The transduction method—Design of logic networks based on permissible functions," *IEEE Trans. Computers,* vol. 38, pp. 1404–1423, Oct. 1989.
[12] W. Kunz and D. Stoffel, *Reasoning in Boolean Networks—Logic Synthesis and Verification Using Testing Techniques.* Norwell, MA: Kluwer, 1997.
[13] J. Rajski and J. Vasudevamurthy, "Testability preserving transformations in multi-level logic synthesis," in *Proc. Intl. Test Conf.,* Washington DC, 1990, pp. 265–273.

[14] B. Rohfleisch and F. Berglez, "Introduction of permissible bridges with application to logic optimization after technology mapping," in *Proc. EDAC/ETC/EUROASIC*, Feb. 1994, pp. 87–93.
[15] R. K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. R. Wang, "MIS: Multi-level interactive logic optimization system," *IEEE Trans. Computer-Aided Design*, vol. 6, pp. 1062–1081, Nov. 1987.
[16] University of California, Berkeley, 1994, "Design Technology Warehouse," [Online]. Available: http://www-cad.eecs.berkeley.edu/Software/software.html.
[17] G. De. Micheli, *Synthesis and Optimization of Digital Circuits.* New York: McGraw-Hill, Inc., 1994.
[18] A. Krasniewski, "Can redundancy enhance testability?," in *Proc. Intl. Test Conf.*, Washington, DC, 1991, pp. 483–491.
[19] A. Krasniewski and A. Albicki, "Random testability of redundant circuits," in *Int. Conf. Computer Design*, Boston, MA, 1991, pp. 424–427.
[20] N. Touba and E. J. McCluskey, "Automated logic synthesis of random pattern testable circuits," in *Proc. Int. Test Conf.*, Washington, DC, 1994, pp. 174–183.
[21] C. H. Chiang and S. K. Gupta, "Random pattern testable logic synthesis," in *Int. Conf. Computer-Aided Design*, San Jose, CA, 1994, pp. 125–128.
[22] P. H. Bardell, W. H. McAnney, and J. Savir, *Built-in Test for VLSI: Pseudo-random Techniques.* New York: Wiley, 1987.
[23] M. J. Batek and J. P. Hayes, "Test-set preserving logic transformations," in *Proc. Design Automation Conf.*, Anaheim, CA, 1992, pp. 454–458.
[24] P. H. Bardell, "Analysis of cellular automata used as a pseudo-random pattern generators," in *Proc. Int. Test Conf.*, Washington, DC, 1990, pp. 762–768.
[25] F. Brglez and H. Fujiwara, "A neural netlist of ten combinational benchmark circuits and a target translator in FORTRAN," in *Proc. Int. Symp. Circuits and Systems*, London, U.K., June 1985, pp. 356–360.
[26] S. K. Jain and V. D. Agrawal, "Statistical fault analysis," *IEEE Design Test Comp.*, vol. 2, no. 1, pp. 38–44, 1984.
[27] S. C Seth, L. Pan, and V. D. Agarwal, "PREDICT—Probabilistic estimation of digital circuit testability," *IEEE Trans. Computers*, vol. C-18, pp. 457–459, 1986.
[28] S. Gupta, "Synthesis of testable combinational circuits: An overview of university activities," presented at International Test Conference, Test Synthesis Seminar, Washington DC, 1994.
[29] T. Sasao, Ed., *Logic Synthesis and Optimization.* Norwell, MA: Kluwer, 1993.
[30] T. Sasao and P. Besslich, "On the complexity of MOD-2 sum PLA's," *IEEE Trans. Computers*, vol. 39, no. 2, pp. 171–178, Feb. 1990.
[31] D. K. Pradhan, "Universal test sets for multiple fault detection in AND-EXOR arrays," *IEEE Trans. Computers*, vol. C-27, pp. 181–187, Feb. 1978.
[32] M. Abramovichi, M. A. Breuer, and A. D. Friedman, *Digital Testing and Testable Design.* New York: Instit. Electrical and Electronics Engineers,, 1990.
[33] L. Hsuiman and V. Iyengar, "Clarifying statistical fault analysis," *IEEE Design and Test*, vol. 3, pp. 101–115, Aug. 1985.
[34] A. Sarabi, "Design of testability properties of AND/XOR networks," in *FIP WG 1 0.5 Workshop on Applications on Reed—Muller Expansion in Circuit Design*, Germany, 1993, pp. 138–142.
[35] J. Saul, "An algorithm for the multi-level minimization of Reed–Muller representations," in *Proc. Int. Conf. Computer Design*, 1991, pp. 634–637.
[36] J. Hartmann and G. Kemnitz, "How to do weighted random testing for BIST," in *Int. Conf. Computer-Aided Design*, Santa Clara, CA, 1993, pp. 568–571.
[37] J. Bell, "Timing analysis of logic level digital circuits using uncertainty analysis," M.S. thesis, Dept. of Computer Science, Texas A&M University, Aug. 1996.
[38] S. Devadas and S. Malik, "A survey of optimization techniques targeting low power VLSI circuits," in *Proc. Design Automation Conf.*, San Francisco, CA, 1995, pp. 242–247.
[39] I. Pomeranz and S. M. Reddy, "Testability considerations in technology mapping," in *Proc. 3rd Asian Test Symp.*, 1994, pp. 151–156.
[40] S. M. Reddy, "Easily testable realizations for logic functions," *IEEE Trans. Computers*, vol. C-21, pp. 1183–1188, Nov. 1972.
[41] H. Hengster, R. Drechsler, and B. Becker, "On local transformations and path delay fault testability," *J. Electron. Test. Theory Applicat.*, vol. 7, pp. 173–192, 1995.
[42] L. Berman and L. Trevillyan, "Global flow optimization in automated logic design," *IEEE Trans. Computer-Aided Design*, vol. 7, no. 6, pp. 557–564, May 1991.

Mitrajit Chatterjee received the B.Tech. degree in electronics and electrical communication engineering from the Indian Institute of Technology, Kharagpur, in 1992 and the M.S. and Ph.D. degrees in computer science, from Texas A&M University, College Station, TX, in 1994 and 1997, respectively.

Among other places, he has worked at AT&T Bell Laboratories, Princeton, NJ; Max Planck Institute, Germany; Synopsys Inc., Beaverton, OR; and Reliable Computer Technology, College Station, TX. He is currently working at the Design Automation Group in Integrated Device Technology, Inc., Santa Clara, CA. His research interests include VLSI synthesis, design, timing analysis, and testing.

Dr. Chatterjee is the recipient of the Dr. B. C. Roy Memorial Gold Medal at I.I.T. Kharagpur in 1992.

Dhiraj K. Pradhan (S'70–M'72–SM'80–F'88) is currently a Visiting Professor in the Department of Electrical Engineering, Stanford University, CA, and is also the COE Endowed Chair Professor in Computer Science at Texas A&M University, College Station, TX. Prior to joining Texas A&M, he served as Professor and Coordinator of Computer Engineering at the University of Massachusetts, Amherst. He also has worked at Oakland University, MI; University of Regina, Canada; and Stanford University, CA. His research interests include VLSI CAD and test, fault-tolerant computing, computer architecture, and parallel processing. Currently, he is also an Editor for several journals, including the IEEE TRANSACTIONS ON COMPUTERS, IEEE PRESS, and the *Journal of Electronic Testing: Theory and Applications (JETTA)*. He is co-author and editor of various books, including *Fault-Tolerant Computing: Theory and Techniques, Vols. I and II* (Englewood Cliffs, NJ: Prentice Hall, 1986), *Fault-Tolerant Computer Systems Design* (Englewood Cliffs, NJ: Prentice Hall, 1996), and *IC Manufacturability: The Art of Process and Design Integration* (Piscataway, NJ: IEEE Press, 1996).

Dr. Pradhan is the recipient of the Humboldt Distinguished Senior Scientist Award, Germany. He is also the recipient of the 1997–1998 Fulbright-Flad Chair in Computer Science. He has also served as General Chair and Program Chair for various major conferences. He has received several Best Paper Awards, including The 1996 IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS Best Paper Award. He has served as Guest Editor of special issues in prestigious journals such as IEEE TRANSACTIONS ON COMPUTERS.

Wolfgang Kunz (S'90–M'91) was born in Saarbrüken, Germany, in 1964. He received the Dipl.-Ing degree in electrical engineering from the University of Karlsruhe, Germany, in 1989 and the Dr.-Ing. degree in electrical engineering from the University of Hannover, Germany, in 1992.

In 1989, he was a Visiting Scientist at the Norwegian Institute of Technology, Trondheim, Norway. From 1989 to 1991, he was with the Department of Electrical and Computer Engineering at the University of Massachusetts, Amherst, and from 1991 to 1993, he was with the Institut für Theoretische Elektrotechnik at the University of Hannover. From 1993 to 1996, he was with the Max-Planck-Society, Fault-Tolerant Computing Group, at the University of Potsdam, Germany, and from 1996 to 1998 he was with the Institute of Computer Science at the University of Potsdam. He has held summer appointments as a Research Assistant Professor with Texas A&M University in 1995 and 1996. Since March 1998, he has been a Professor with the Department of Computer Science, University of Frankfurt, Germany. His research interests are in VLSI testing, formal verification, logic synthesis, and fault-tolerant computing.

Dr. Kunz is recipient of the Berlin-Brandenburg Academy of Sciences Award and IEEE TRANSACTIONS ON CAD Best Paper Award.