ATPG-based Transformations for Random-Pattern Testable Logic Synthesis [1]

M. Chatterjee D.K. Pradhan W. Kunz

Dept. of Computer Science
Texas A & M University
College Station, TX-77843

Abstract

A new approach to designing random pattern testable logic circuits is introduced. The synthesis methodology proposed here begins with a multi-level circuit and synthesizes the circuit, optimizing its area and enhancing its random pattern testability. The method is based on ATPG-based transformations. A scheme is presented to guide the synthesis process and introduces certain new transformations which enhance the sensitivity of the circuit. Previous approaches either do not consider random pattern testability, or enhance random pattern testability only when beginning from a two-level circuit representation. The proposed method can take these synthesized multi-level circuits as inputs, further processing them to enhance random testability. In fact, the experimental results demonstrate that the proposed method can be successfully applied on circuits synthesized by a logic synthesis tool (tstfx), with a large reduction in area as well as a gain in random testability. It is possible that the new transformations introduced in the paper can cover new search spaces during optimization, thus aiding in area optimization.

[1] Research reported is supported by NSF grant MIP 9406946.

1 Introduction

Traditional goals in all automatic synthesis and optimization of multi-level combinational circuits [13, 11, 10, 8, 7, 9, 22] include minimization of area and performance. While, most of these tools ensure that all faults of interest are testable, *random testability* [1, 21, 24] of the faults is not considered. As built-in-self-test (BIST) has been widely accepted for testing complex circuits, it is helpful if the goals of the automated logic synthesis procedure also include *random testability.* Until recently, techniques were proposed to design multi-level logic circuits keeping BIST in mind [17, 18]. But, both these approaches begin from a two-level representation of the circuit and synthesize a multi-level circuit. Techniques which take a multi-level circuit as the starting circuit representation and *improves* the random testability based on logic transformations have not yet been studied and warrants more investigation. This paper presents a methodology for optimizing multi-level logic circuits based on transformations that enhance random testability. Discussed here is random testability with respect to *s-a-faults.*

BIST has been widely adopted for its capability to relieve the difficulties of using an external tester, yielding lower production cost [1]. However, extra hardware and performance penalty can be incurred for implementing a high fault-coverage self-test mechanism. Typically, in BIST, test vectors are generated on the chip by using hardware test pattern generators (TPG) such as linear feedback shift registers (LFSR)[1], or by cellular automata [20]. It is crucial to have as few test patterns as possible to achieve acceptable test coverage. Not all faults can be tested easily in BIST. Those faults that need long TPG sequences for testing are termed as *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, include :

- Apply biased test patterns [1, 2, 5].
- Embed test vectors on the chip [3, 4, 6].
- Insert controllability or observability test points [25].

These techniques use complex additional hardware, either inside the circuit with additional pins (test points) or in the TPG. The additional hardware incurred can only be miminized if the circuits are synthesized with a higher random pattern testability.

The paper proposes a methodology of multi-level circuit synthesis which combines area optimization with random pattern testability. The proposed technique begins with a multi-level circuit and applies logic transformations that contribute to reducing the area and enhancing random testability. The transformations used here are of two types :
(a) *ATPG-based* transformations [7].
(b) Transformations increasing the sensitivity of the circuit.
These transformations are used iteratively to yield an optimized circuit. The resultant circuits were evaluated by random pattern simulation based on LFSRs.

The paper is organized as follows. The next section porovides a brief summary of previous work. Section 3 describes the proposed synthesis methodology. Section 4 proposes certain transformations which introduce XOR gates in the circuit. Section 5 presents experimental results on PLA benchmarks. Conclusions appear in Section 6.

2 Previous Work

Previous research in random testable logic synthesis consisted of redundancy removal from combinational circuits [28], synthesis of fully testable circuits [11, 14] and synthesis of random testable circuits from two-level circuits [17, 18].

Testability-preserving transformations involving multiple stuck at faults have been studied in [29]. It has been proposed in [15, 16] that careful assignment of don't cares of functions and redundant lines can improve the detectability profile of circuits. The logic synthesis procedure in [11] is based on extraction of single and double cubes which have been shown to preserve testability of single s-a-fault. Transformations are identified [14] where the synthesized logic will have a reduced deterministic test length.

The two previous random pattern testable circuit synthesis methods [17, 18] starting with a two-level circuit, first determines the random testability of the circuit and identifies the HTD faults. The transformations in [17] are limited to algebraic factors, namely kernels and common cubes. The algebraic transformations used in [18] include *single cube division*, *double cube division* and some *double cube divisions with multiple outputs.* In both methods, each transformation is evaluated based on its impact on the random testability of the circuit which guides the synthesis procedure. The testability preserving transformations proposed in [11, 14] are also used in [17] where test points are inserted based on fault detection probabilities.

3 Proposed Method

The proposed method can take as its input a multi-level circuit and synthesizes a randomly testable circuit with a lesser area. As mentioned, the proposed synthesis method uses *ATPG-based transformations* [7] as well as certain additional transformations which increase the sensitivity of the circuit. Transformations of the former type are chosen based on their capability to enhance random testability. The random testability of the circuit is estimated at every iteration to give a proper guidance to the synthesis process. This section first gives the basic preliminaries and reviews *ATPG-based transformations* [7] used in our synthesis process. Propose is our method of *incremental* testability estimation and how it guides the synthesis procedure. Transformations based on XOR gates that increase the sensitivity of the circuit are proposed in the next section.

3.1 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 XOR 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 $x_1, ..x_n$ 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 $C \equiv C'$, 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* 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 non-equivalent function y' if this does not change the function $C(\vec{x})$:$B_2^n \rightarrow B_2^m$ of the combinational network as a whole. Such functions are called *permissible functions* [10].

3.2 ATPG-Based Transformations - Review

ATPG-based transformations have been chosen to drive our synthesis procedure because of two main reasons :

- The transformations are simple and testability based on these transformations can be estimated with low time overhead.
- The ATPG-based expansions can be used to perform arbitrary manipulations in a combinational network [7], thus covering almost all other traditional logic transformations.

ATPG-based transformations are based on the well-known Shannon expansion. For an arbitrary boolean function, $f(\vec{x})$:$B_2^n \rightarrow B_2$ and $y(\vec{x})$:$B_2^n \rightarrow B_2$ with $B_2 = \{0,1\}$ the following equation holds:

$$y(\vec{x}) = f(\vec{x}).y(\vec{x}) \mid_{f(\vec{x}=1)} + \bar{f}(\vec{x}).y(\vec{x}) \mid_{f(\vec{x}=0)} \tag{1}$$

short notation: $y = f.y \mid_1 + \bar{f}.y \mid_0$

The methodology suggested can be viewed as a two-step process :

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

The first step is equivalent to performing a *division* [22]. Combining it with ATPG-based redundancy elimination results in an expansion that can also be seen as special *transductions* (transformation and reduction) [10]. The method to identify divisors is based on *indirect implications* [19]. If a value assignment at a node y allows us to imply a unique value assignement at node f, then equation 1 can be simplified, as stated in the following lemmas [7]. The node f must not be in the transitive fanout of y, to ensure that the circuit remains combinational after the transformation.

Lemma 3.1: Consider the transformation T_1: $y' = y \mid_1 + \bar{f}$. Then $y' = y$ if and only if the implication $y = 0 \Rightarrow f = 1$ is true [7].

Lemma 3.2: Consider the transformation T_2: $y' = y \mid_0 + f$. Then $y' = y$ if and only if the implication $y = 0 \Rightarrow f = 0$ is true [7].

Lemma 3.3: Consider the transformation T_3: $y' = y \mid_1 .f$. Then $y' = y$ if and only if the implication $y = 1 \Rightarrow f = 1$ is true [7].

Lemma 3.4: Consider the transformation T_3: $y' = y \mid_0 .\bar{f}$. Then $y' = y$ if and only if the implication $y = 1 \Rightarrow f = 0$ is true [7].

Example 3.1: Take the circuit shown in Fig1(a) as an example. It can be observed that $y = 0 \Rightarrow f = 0$ and transformation T_2 can be applied on the circuit. The resultant circuit is shown

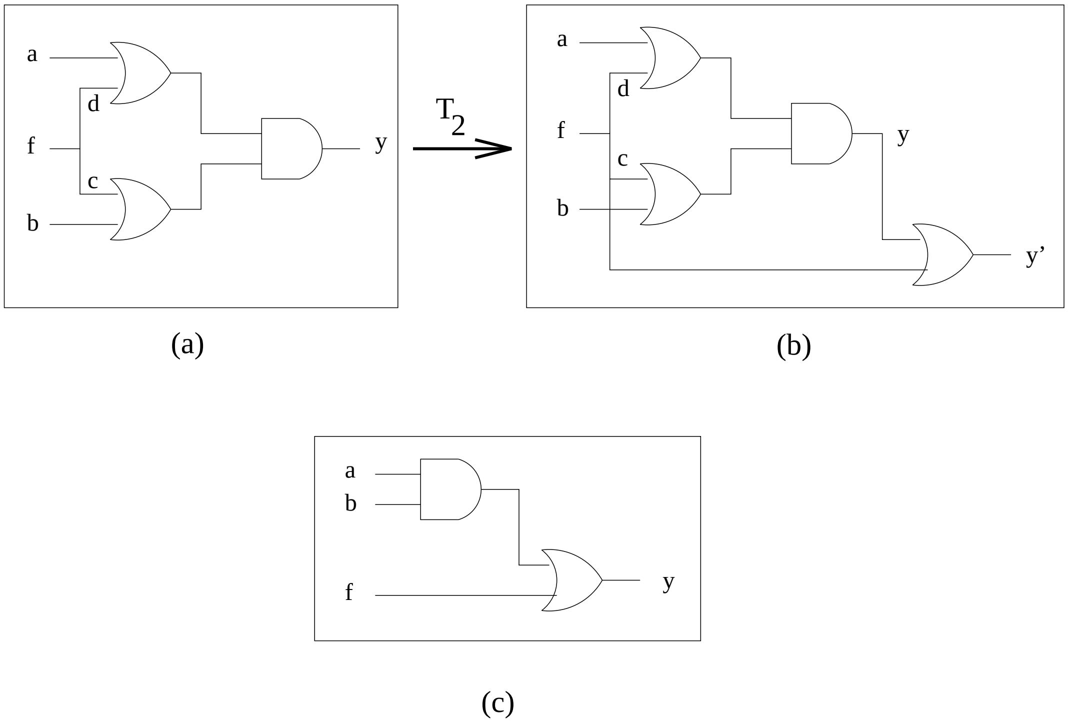

Figure 1: *An example of an ATPG-based transformation.*

in Fig 1(b). ATPG processing on the resultant circuit would determine lines 'c' and 'd' to be as redundant s-a-1 lines and hence, can be eliminated. The final circuit after redundancy elimination is shown in Fig 1(c).□.

Transformations identified by 'D-implications' are related to *permissible functions*. If $f = U$, $U \in 0,1$ is a value assignment at at node f which is *necessary* to detect the fault, y s-a-V, at at least one primary output of the combinational network C, then $y = \bar{V}$ 'D-implies' $f = U$ and is denoted by: $y = \bar{V} \xrightarrow{D} f = U$. The conventional implications can be viewed as a special case of such D-implications. The transformations defined in [7] are based on the following four theorems :

Theorem 3.1: (Analogous to *Lemma 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 both stuck-at faults at node y are testable. The function y':$B_2^n \rightarrow B_2^m$, $B_2 = \{0,1\}$ with $y' = y \mid_1 + \bar{f}$ is a permissible function at node y iff the D-implication $y = 0 \xrightarrow{D} f = 1$ is true [7].□

Theorem 3.2 - 3.4: Analogous to *Lemmas 3.2 - 3.4*.

D-implications to be used in the transformations are determined by *recursive learning* [19]. This is accomplished by two routines: *make_all_implications()* and *fault_propagation_learning()*, as given in [19], if they are performed to the Roth's five-valued logic alphabet. The ATPG-based synthesis procedure only looks at *indirect implications* as the promising candidates for optimization.

3.3 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 [1, 2], 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-observatbility) of the line. The 1-observability (0-observability) of a line y is the probaility 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 it is connected. The U value of an output line or a fanout stem is defined as 1. Simulations are done for the good machine to estimate the controllabilities and the U values of each line in the circuit. The method of calculating the observabilities from the estimated parameters is described in [26, 1].

The proposed method begins by applying STAFAN to estimate the fault detection probabilities and identify the HTD faults in the given multi-level 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 CO values and the U value of each node in the circuit, based on the ATPG-based transformations.

As the ATPG-based transformation is a two step process, transformation and reduction, updates on the estimated parameters also are done in two steps. For the *first step*, transformations are based on Theorems 3.1 - 3.4. 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**.

The procedure first sets the CO_1 (CO_1) values of the s-a-1 (s-a-0) redundant lines equal to 1. This would create a change in the CO values of the 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,..,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,..,a_k. If no-fanout exists in the circuit, $CO_1(b)$ can be estimated by the following equation :

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

```
    Update_Stafan(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
```

Figure 2: Algorithm to estimate CO values and U values during Redundancy Elimination

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.

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

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 on the nature of the 'implication' relationship on which it is based. Given below is the description of the update procedure for the transformation-based on Theorem 3.1. The update procedures based on Theorems 3.2 - 3.4 can be implemented similarly.

Consider the subcircuit shown in Fig 3(a). A transformation based on the D-implication, $y = 0 \xrightarrow{D} f = 1$ (Theorem 3.1) would result in the subcircuit shown in Fig 3(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, $f1$, $f2$, $f3$ and y'. The updates values of the CO values and the U values are expressed as :

$$\begin{gathered} CO_1(f_1) = CO_1(f_2) = CO_0(f_3) = CO_1(f) \\ CO_0(f_1) = CO_0(f_2) = CO_1(f_3) = CO_0(f) \\ CO_0(y') = CO_0(y);\ CO_1(y') = CO_1(y) \end{gathered}$$

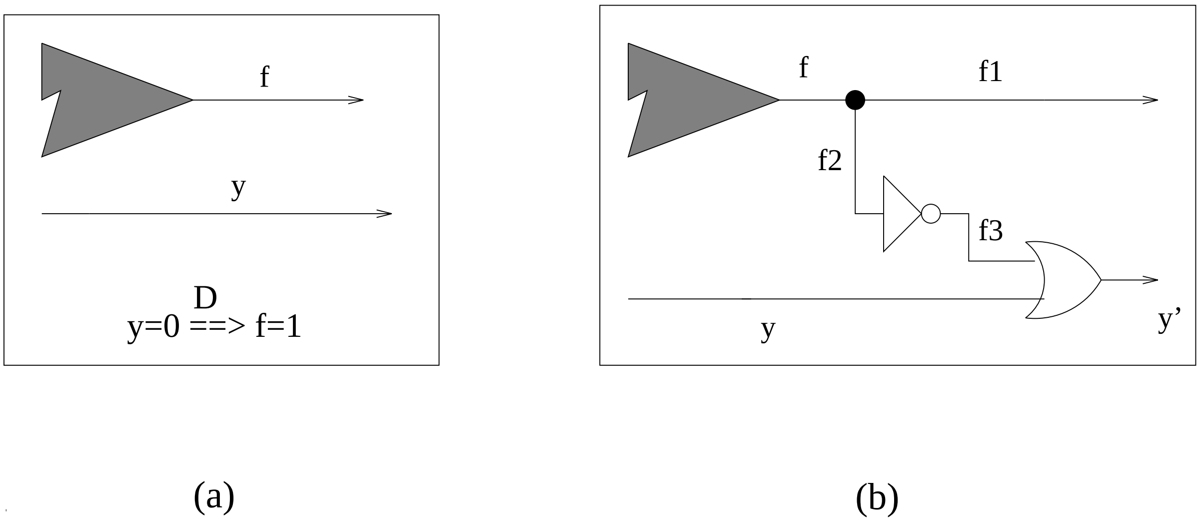

Figure 3: *Transformation based on Theorem 3.1*

$$U(f_1) = U(f);\, U(f_3) = CO_1(y)$$
$$U(y') = U(y);\, U(y) = CO_1(f)$$

Because these estimates are determined regardless of whether the transformation was based on an equivalent function or a 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

Estimating the fault detection probabilities (PDs) of a transformed circuit evaluates the cost of the transformation with regard to random pattern testability. The testability cost (TC) takes into account the change in the fault detection probabilities of the HTD faults, as these faults cause an effect on the random pattern test length. In the proposed method, HTD faults are those faults whose fault detection probailities lie between PD_min and $(PD\_min \times PD\_range)$. PD_min is the PD of the node with the minimum fault detection probability. PD_range is a predetermined value, and depends on the nature of the circuit. Thus, the TC of a transformation is given by :

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

3.4 Choosing ATPG-based Transformations for Random Testability

ATPG-based transformations which enhance random testability can be classified into two types :

1. ATPG-based transformation enhancing testability with no increase in area.
2. ATPG-based transformation enhancing testability with increase in area.

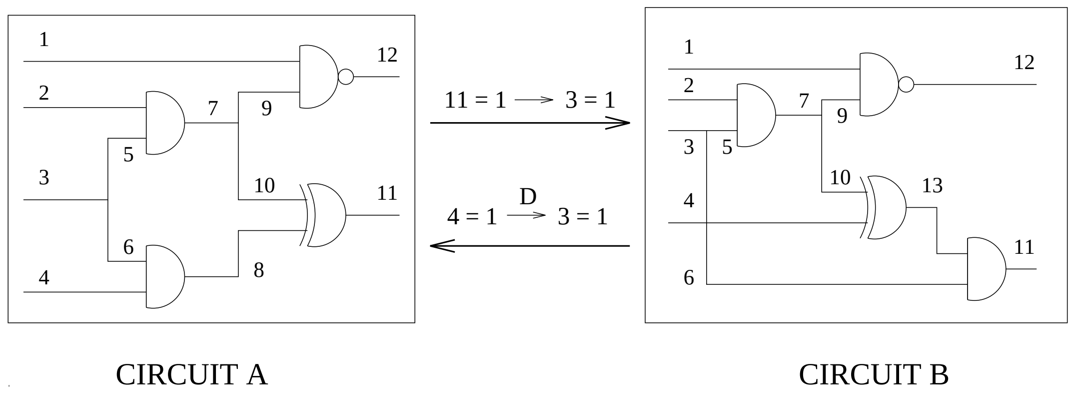

Figure 4: *ATPG-based Transformations for Example 3.1*

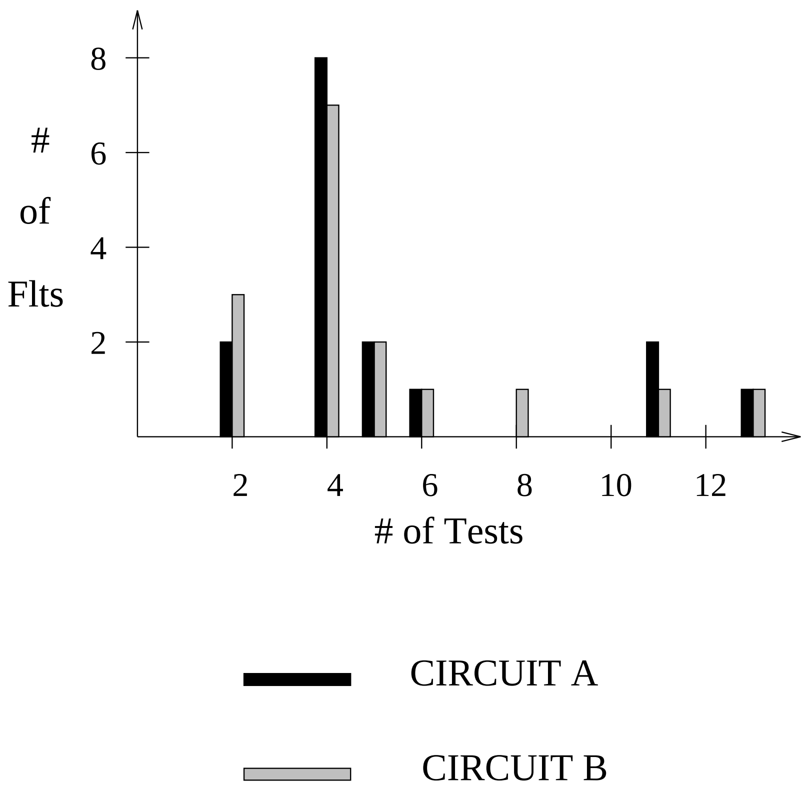

Figure 5: *Detectability Profile*

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 [14]. 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 3.1: Consider two circuits, A and B in Fig 4. Circuit A can be transformed to Circuit B using the transformation based on (line 11 = 1) $\rightarrow$ (line 3 = 1). Furthermore, circuit B can be transformed back to circuit A using the transformation (line 4 = 1) $\xrightarrow{D}$ (line 3 = 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. 5. 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. If the circuit is a sub-circuit of a larger circuit, the test vectors at the input of the sub-circuit may not be equiprobable and the random testability might be different.□

ATPG-based transformations associated with no increase in area are the area-optimization transformations used in [7]. In the proposed method, among these, only those transformations are accepted where the estimates *testability cost* (equation 5) are positive. Thus, area optimization is allowed with improvement of random pattern testability of the circuit.

ATPG-based transformations associated with increase in area involve addition of redundant lines which improve the *observability* of areas in the circuit containing a high concentration of HTD faults. Addition of redundant lines will be based on Theorems 3.1-3.4. Figure 3 gives the example of a transformation based on Theorem 3.1. 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 have a large concentration of HTD faults and where y has a high observability are useful in enhancing random pattern testability. The proposed synthesis tries to introduce a large number of such redundant lines and ensures that these lines are not removed in the next few iterations.

The two types of transformations are used iteratively - one after the other. The transformations of the first kind try to maintain a low area without degrading the random testability. The other one improves random testability, and increases the area of the circuit from time to time. Increasing the area might also help the area optimization procedure come out of a local minima [22, 7, 9].

4 Transformations Involving XOR Gates

It is observed that circuits containing a large number of XOR gates have a high random testability. This is because outputs of XOR gates are not biased towards a particular value, and the output of an XOR gate is sensitive to all its inputs. Thus, transformations introducing XOR gates in a combinational circuit will always enhance single s-a-fault random testability. This section, proposes transformations which introduce XOR gates as a new type of ATPG-based transformation which will enhance random pattern testability. We propose three such types of transformations to enhance random pattern testability.

4.1 Transforming Gate Functionality

Two-input gates in circuits can be transformed to XOR gates or XNOR gates if they are *permissible*, based on ATPG assignments. A two-input gate, when transformed to an XOR gate, does not immediatley enhance the random testability . If the transformation is followed by some other transformations, the XOR gate may play an active role in random testability enhancement. A two-input AND (OR) gate or a two-input NOR (NAND) gate can be transformed to an XNOR (XOR) gate if the {0,0} ({1,1}) input is an observability don't care [22, 9]. Identifying such transformation using ATPG is done by the following theorems:

Theorem 4.1: Let y be a node feeding from an AND (NOR) gate with two inputs, a_0 and a_1. If the necessary condition to test for y s-a-1 satisfies ($a_0 \oplus a_1 = 1$), the AND (NOR) gate can be replaced by a *permissible* XNOR gate.

Proof: The input combinations required to test a s-a-1 at the output of an AND (NOR) gate are {0,0}, ({1,1}), {0,1} and {1,0}. When the necessary condition satisfies the relation ($a_0 \oplus a_1 = 1$), the input combination {0,0} ({1,1}) to the AND (NOR) gate can be treated as a don't care to the circuit behaviour. □

Theorem 4.2: Let y be a node feeding from an OR (NAND) gate with two inputs, a_0 and a_1. If the necessary condition to test for y s-a-0 satisfies ($a_0 \oplus a_1 = 1$), the OR (NAND) gate can be replaced by a *permissible* XOR gate.□

The search for such gate functionality transformations can be used iteratively between the other ATPG-based transformations.

4.2 Transforming with XOR Gates Based on D-Implication

The ATPG-based transformations proposed earlier introduce either AND or OR gates in the circuit. XOR gates can be introduced as *permissible* functions when D-implications for a line y stuck-at-0 require an assignment which is the same as that for the line y stuck-at-1. The following theorem formally defines the condition for introducing *permissible* XOR 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 both stuck-at faults at node y are testable. The function y':$B_2^n \rightarrow B_2^m$, $B_2 = \{0,1\}$ with $y' = y\,|_1 \oplus f$ is a permissible function at node y iff the D-implications $y = 0 \xrightarrow{D} f = 0$ and $y = 1 \xrightarrow{D} f = 0$ 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 both stuck-at faults at node y are testable. The function y':$B_2^n \rightarrow B_2^m$, $B_2 = \{0,1\}$ with $y' = y\,|_1 \oplus \bar{f}$ is a permissible function at node y iff the D-implications $y = 0 \xrightarrow{D} f = 1$ and $y = 1 \xrightarrow{D} f = 1$ are true.□

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 XOR gate may result in more than one line becoming redundant, thus aiding area optimization. The XOR gate will also enhance the observability of all the faults in the input cone of, f resulting in better random testability.

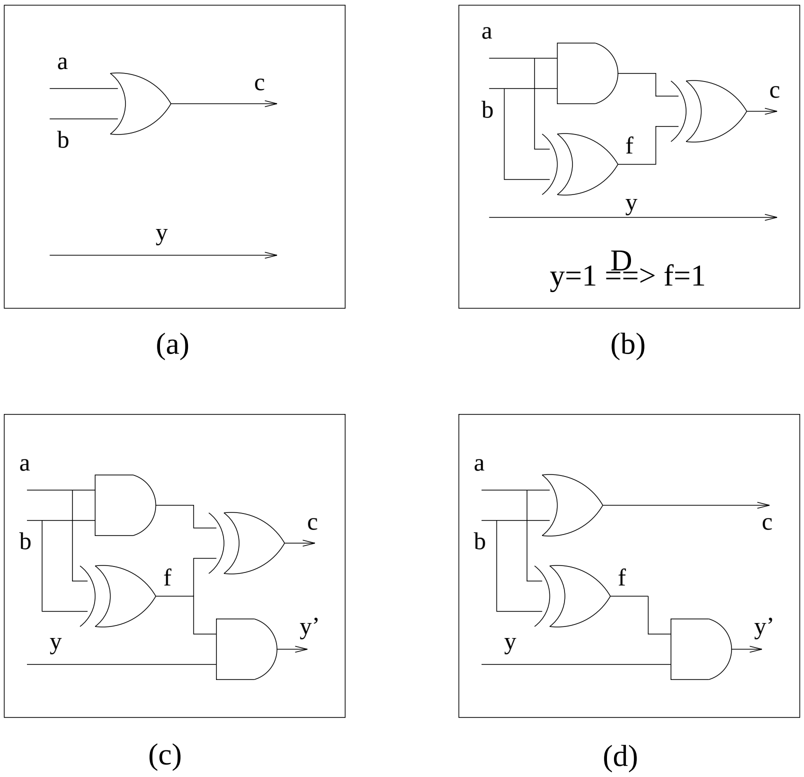

Figure 6: *Example of ATPG-based transformations using Reed-Muller expansions.*

4.3 Transformations 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' for ATPG-based transformations. Two-input AND gates and OR gates can be expanded by the relations

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

To represent the expanded relations, the output of the new gates introduced can become a part of an ATPG-based transformation. These transformation are different from those used in [7] and hence, aid in circuit optimization and random testability enhancement. Given below is an example to give better understanding to these types of transformations.

Example 4.1: Consider a sub-circuit shown in Fig 6(a). Let the OR gate in the sub-circuit be expanded, based on its Reed-Muller expansion as shown in Fig 6(b). Now, it may happen that in the sub-circuit, there is a D-implication denoted by $y = 1 \xrightarrow{D} f = 1$. The circuit transformation based on the D-implication is shown in Fig 6(c). The final sub-circuit, after extracting back the OR gate, is in Fig 6(d). The transformation from the sub-circuit in Fig 6(a) to the one in Fig 6(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 1. This would enhance the detection probabilities of all the faults in the fanin cone of a and b.$\Box$.

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

Circuit	I/O	Literals			Connections		
		tstfx	*Proposed*	*%imp*	*tstfx*	*Proposed*	*%imp*
b10	15/11	521	467	10.3	764	653	14.5
b4	33/23	362	354	1.6	499	481	3.6
gary	15/11	573	501	12.5	847	682	19.4
in2	19/10	507	468	7.7	735	655	10.9
in4	32/20	616	438	28.9	880	588	33.2
in5	24/14	333	305	8.1	454	389	14.3
in6	33/23	372	341	8.3	512	452	11.7
in7	26/10	148	109	26.3	199	137	31.3
vg2	25/8	241	124	48.5	353	170	51.8
x1dn	27/6	352	122	65.3	501	171	65.8

Table 1: Comparison of circuits for the proposed technique with *tstfx*.

5 Experimental Results

Experiments were conducted on the PLA benchmark circuits to evaluate the performance of the proposed methods. The synthesis methodology proposed in Section 3 was implemented. The synthesis tool takes in as its input circuits synthesized by *tstfx* [18] and optimizes the circuit which enhances random testability. Transformations based on XOR gates (Section 4) are currently being implemented and to be subsequently published later.

To make the circuits compatable to our implementation, the multi-level circuits synthesized by *tstfx* were mapped to a SIS library obtained from MCNC.genlib, they contain no complex gates. The mapped circuits were then synthesized by the proposed synthesis tool. For the sake of comparison, these circuits were also synthesized by HANNIBAL [7], which only optimizes area. The random testability of all the circuits was obtained by using LFSR-generated pseudo-random sequences. The test sequences presented are an average, over several different LFSR-generated sequences (typically 5-10 primitive polynomials were used). The circuits synthesized by the proposed method are compared to those synthesized by *tstfx* and to those synthesized by HANNIBAL.

5.1 Comparison with *tstfx*

The circuits synthesized by *tstfx* and the proposed method are compared, based on their area and the random pattern testability of the circuit. The area comparison is presented in Table 1. The area is compared by literal count and the number of connections in the circuit. It can be observed that area of the *tstfx* synthesized circuits was further reduced by 25.56%, on the average, by the proposed synthesis methodology.

The comparison of the random pattern testability is shown in Table 2. The second column gives the *PD_range* used in the proposed method to classify the HTD faults. The fault coverage considered for each circuit is given in Column 3. It was found that except for one circuit (in6), the circuit synthesized by the proposed method is better with respect to area and random pattern

Circuit	PD_range	Fault Coverage	Test Length		Percentage Improved
			tstfx	*Proposed*	
b10	10	98%	8020	7521	6.2
b4	3	99%	26240	24960	0.0
gary	4	98%	8864	7664	13.5
in2	2	98%	5620	4066	27.6
in4	50	98%	37543	9025	75.9
in5	5	98%	5600	5137	8.26
in6	3	98%	13632	13756	-0.0
in7	10	98%	10962	8384	23.5
vg2	2	95%	36989	10026	72.8
x1dn	10	92%	51000	13216	74.1

Table 2: Comparison of random pattern testability for the proposed technique with *tstfx*.

Circuit	I/O	Literals			Connections		
		HANNIBAL	*Proposed*	*%imp*	*HANNIBAL*	*Proposed*	*%imp*
b10	15/11	428	467	-8.3	560	653	-14.2
b4	33/23	281	354	-20.6	358	481	-25.0
gary	15/11	444	501	-11.3	575	682	-15.6
in2	19/10	276	468	-41.0	360	655	-45.0
in4	32/20	304	438	-30.5	394	588	-31.9
in5	24/14	258	305	-15.4	318	389	-18.2
in6	33/23	293	341	-14.7	359	452	-20.0
in7	26/10	93	109	-14.6	117	137	-14.5
vg2	25/8	89	124	-18.5	124	170	-27.0
x1dn	27/6	99	122	-18.8	138	171	-19.2

Table 3: Comparison of circuits for the proposed technique with HANNIBAL.

Circuit	PD_range	Fault Coverage	Test Length		Percentage Improved
			HANNIBAL	*Proposed*	
b10	10	98%	8101	7521	7.1
b4	3	99%	50241	24960	50.1
gary	4	98%	9256	7664	17.1
in2	2	98%	2505	4066	-38.4
in4	50	98%	14240	9025	36.6
in5	5	98%	5578	5137	7.9
in6	3	98%	21538	13756	36.1
in7	10	98%	10962	8384	23.5
vg2	2	95%	15226	10026	34.1
x1dn	10	92%	20610	13216	35.6

Table 4: Comparison of random pattern testability for the proposed technique with HANNIBAL.

test length. The reduction in random pattern test length compared to *tstfx* is, on the average, 30.4%.

The circuits synthesized by *tstfx* [18] are more randomly testable compared to the circuits synthesized by SIS[13]. As the circuits synthesized by the proposed method are more randomly testable and area optimized than the ones synthesized by *tstfx*, they are more randomly testable and comparable in area to the circuits synthesized by SIS.

5.2 Comparison with HANNIBAL

Comparison of the proposed method with HANNIBAL (Table 3) shows that the proposed method, with its present implementation, does sacrifice some area to gain in random pattern testability. The areas of these circuits is found to be larger than those by HANNIBAL, by around 20% on the average. This is because the proposed algorithm sometimes become too restrictive when allowing ATPG-based transformations during optimization.

Comparison of the random pattern testability, in Table 4, shows that for all circuits except in one (in2), the proposed method outperforms HANNIBAL in synthesizing randomly testable circuits. The average improvement of random pattern test length is 22%.

It can be observed that the circuits synthesized by the proposed method is more randomly testable than those by HANNIBAL, but sacrifices in area compared to HANNIBAL, as HANNIBAL only concentrates on area optimization.

6 Conclusions

A novel optimization procedure using recursive learning based implications is presented here. The transformations proposed are based on (a) heuristics to identify difficult-to-test faults (b) methods to guide the synthesis process enhancing random pattern testability and (c) new logic transformations based on XOR gates. The transformations used in the proposed scheme iteratively increaces and decreases the area of the circuit for exploring different search spaces. The new XOR logic transformations are developed with the objective of enhancing observability difficulties which is the primary source of low detection probability. The synthesis scheme can be applied to poor random-testable multi-level circuits which were synthesized by another synthesis tool to improve random testability as well as area. Experimental indicate improvement over other optimization tools including those designed specifically for testable synthesis in terms of improved testability and/or area. Currently, the new transformations introduced in the paper is being studied in the context of area optimization.

References

[1] P. H. Bardell, W.H. McAnney, J. Savir, *Built-in test for VLSI : pseudo-random techniques*, John Wiley and Sons, New York 1987.

[2] M. Abramovichi, M.A. Breuer, A.D. Friedman, "Digital Testing and Testable Design," Compter Science Press.

[3] V.K. Agarwal, E. Cerny, "Store and Generate Built-in Testing Approach," Proc. FTCS 11, pp. 35-40, June 1981.

[4] S.J. Upadhyaya, L.C. Chen, "On-chip Test Generation for Combinational Circuits by LFSR Modification, " ICCAD 1993.

[5] S. Pateras, J. Rajski, "Cube-Contained Random Patterns and their Application to the Complete Testing of Synthesized Multi-level Circuits," Intl. Test Conference, pp. 473-481, 1991.

[6] S. Akers, W. Jansz, "Test Set Embedding in a BIST Environment," Intl. Test Conference, pp. 257-263, 1989.

[7] W. Kunz, P.R. Menon, "Multi Level Logic Optimization by Implication Analysis," ICCAD 1994.

[8] K.T. Cheng, L.A. Entrena,"Multi-level Logic Optimization by Redundancy Addition and Removal," EDAC, pp. 373-377, 1993.

[9] S.C. Chang, M. Marek-Sadowska,"Perturb and Simplify: Multi-Level Boolean Network Optimizer," ICCAD, pp. 2-5, 1994.

[10] S. Muroga *et. al*, "The Transduction Method - Design of Logic Networks Based on Permissible Functions," IEEE Trans on Computer, Oct 1989, pp. 1404-1424.

[11] J. Rajski *et. al*, "Testability Preserving Transformations in Multi-Level Logic Synthesis," Intl. Test Conference, pp. 265-273, 1990.

[12] B. Rohfleisch, F. Berglez, "Introduction of Permissible Bridges with Application to Logic Optimization after Technology Mapping," Proc. EDAC/ETC/EUROASIC, 1994.

[13] R.K. Brayton, *et. al*, "MIS: Multi-level Interactive Logic Optimization System," IEEE Trans on CAD, pp. 1062-1081, Nov., 1987.

[14] M.J. Batek, J.P. Hayes, "Test-Set Preserving Logic Transformations," DAC, 1992.

[15] A. Krasniewski, "Can Redundancy Enhance Testability ?," Intl. Test Conference, pp. 483-491, 1991.

[16] A. Krasniewski, A. Albicki, "Random Testability of Redundant Circuits," Intl. Conference on Computer Design, 1991.

[17] N. Touba, E.J. McCluskey, "Automated Logic Synthesis of Random Pattern Testable Circuits," Intl. Test Conference, pp. 174-183, 1994.

[18] C.H. Chiang, S.K. Gupta, "Random Pattern Testable Logic Synthesis," ICCAD 1994.

[19] W. Kunz, D.K. Pradhan, "Recursive Learning: A New Implication Technique for Efficient Solution to CAD Problems - Test, Verification and Optimization," IEEE Trans. on CAD, pp. 1143-1158, Sept., 1994.

[20] P.H. Bardell, "Analysis of Cellular Automata used as a Pseudo-random Pattern Generators," Intl. Test Conference, pp. 762-768, 1990.

[21] E.J. McCluskey,"Verification Testing - a Pseudoexhaustive Test Technique", IEEE Trans. on Compt., vol. C-33, pp. 541-546, June 1984.

[22] G. De. Micheli, "Synthesis and Optimization of Digital Circuits," McGraw-Hill, Inc. 1994.

[23] F. Brglez, H. Fujiwara , "A Neural Netlist of ten Combinational Benchmark Circuits and a Target Translator in FORTRAN,", Proc. Intl. Symp. on Circuits and Systems, June 1985.

[24] Y.K. Malaiya, S. Yang, "A Coverage Problem for Random Testing," Proc. IEEE Intl. Test Conference, pp. 237-245, Nov. 1984.

[25] B. Krishnamurthy, "A Dynamic Programming Approach to the Test Point Insertion Problem," DAC, 1987.

[26] S.K. Jain, V. D. Agrawal, "Statistical Fault Analysis," IEEE Design Test Comp., pp. 38-44, **2**(1).

[27] S.C Seth, *et. al*, "PREDICT - Probabilistic Estimation of Digital Circuit Testability," IEEE Trans. Compt., pp. 457-459, **C-18**(5).

[28] D. Brand, "Redundancy and Don't Cares in Logic Synthesis," IEEE Trans. on Computers, pp. 947-952, C-32(10).

[29] G. D. Hachtel, R. M. Jacoby, K.Keutzer, and C. R. Morrison, "On Properties of algebraic Transformations and Synthesis of Multifault Irredundant Circuits," IEEE Trans. on CAD, vol. 11, pp. 313-321, Mar 1992.