Random Pattern Testable Logic Synthesis

Chen-Huan Chiang and Sandeep K. Gupta

CENG Technical Report 94-08

Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, California 90089-2562
(213)740-2251

April 1994

Random Pattern Testable Logic Synthesis

Chen-Huan Chiang and Sandeep K. Gupta

Electrical Engineering – Systems
University of Southern California
Los Angeles, CA 90089-2562

Abstract

Previous procedures for synthesis of testable logic guarantee that all faults in the synthesized circuits are detectable. However, the detectability of many faults in these circuits can be very low leading to poor random pattern testability. A new procedure to perform logic synthesis that synthesizes circuits that are random pattern testable is proposed. Experimental results show that the circuits synthesized by the proposed procedure tstfx *are significantly more random pattern testable and smaller than those synthesized using its counterpart* fast_extract (fx) *in* SIS. *The proposed synthesis procedure designs circuits that require only simple random pattern generators in built-in self-test, thereby obviating the need for complex BIST circuitry. If external testing is employed, then the computational complexity of test pattern generation and the resulting test set sizes will decrease. Hence, the proposed logic synthesis procedure will decrease the complexity of test problems (with little or no area overhead).*

1 Introduction

Increasing the quality level of shipped products is a prime concern for test engineers. Today, desired reject ratio is 10 or fewer parts per-million for increasingly complex circuits. Such quality level can be achieved only if test considerations are an integral part of the design process. Consequently, design-for-testability (DFT) and built-in self-test (BIST) have achieved wide acceptance. BIST provides a framework to reduce the effort otherwise required to generate extensive test sets for large circuits. It also reduces/eliminates the need for sophisticated test equipment. However, extra hardware test pattern generators (TPGs) are needed and some performance penalty is sustained for implementing BIST. Test vectors are generated on the chip during self-test by using hardware test pattern generators. Typically linear feedback shift registers (LFSRs) or cellular automata (CA) are used as pseudo-random TPGs. On-chip generation of tests enables the application of tests at normal circuit speeds. This allows the application of longer test sequences (for a given test time) than would be possible using a slower test application by an external tester. However, length of pseudo-random sequences required to achieve high fault coverage can be very large. In such cases, complex TPG designs (e.g. weighted random pattern generators) are required to achieve the target fault coverage in acceptable test time. The use of complex TPGs further increases the area and performance penalties of BIST schemes.

Increasingly, logic blocks in commercial circuits are synthesized using logic synthesis tools. The traditional objectives of logic synthesis procedures are minimization of circuit area and delay. Due to the importance of testing, testability is becoming considered during logic synthesis as well. The definition of testability used by most testable logic synthesis procedures is that all faults of interest (e.g. single/multiple stuck-at or path delay faults) are testable. Since all faults are testable in these circuits, the automatic test pattern generator (ATPG) does not have to waste large computational effort in searching for tests for faults that are undetectable.

Unfortunately, while all faults in the circuits designed by these procedures are detectable, many faults in the resulting circuit have few tests. Hence, if external tests are

employed, then ATPG may need to spend large computational effort to find tests for these *hard-to-detect* faults. In BIST, very long LFSR generated pseudo-random sequences may be required to achieve desired fault coverage. Hence, to achieve high fault coverage, in reasonable test time, complex BIST TPG designs are needed thereby increasing the area and performance overheads. The objective of the proposed research is to enhance the notion of testability of circuits to *random pattern testability*. Synthesis procedures to ensure that not only all faults in the synthesized circuit are detectable, but that they are *random pattern testable*, is proposed. If BIST is used, then circuits implemented by the proposed synthesis procedure will require only simple BIST TPGs. Even for external testing, high random pattern testability of faults decreases ATPG effort and test set size. In the following we begin with motivation for the proposed research. A discussion of related research follows. The proposed procedure is outlined in Section 3. Experimental results on PLA benchmark circuits show that the proposed procedure not only results in circuits that are more random pattern testable, but it also reduces the circuit area for most examples. Finally, conclusions and directions for future research are presented.

2 Background

2.1 Motivation

Random pattern testability of several benchmark and other circuits has been studied extensively in the recent years. Typically high coverage of single stuck-at faults ($> 98\%$) can be obtained even for circuits with large number of inputs (> 40) using short (only 1000s of vectors) pseudo-random test sequences generated by LFSRs. On the other hand, it was observed that synthesized circuits required much longer random test sequences to achieve high fault coverage. This observation prompted us to perform the following experiments.

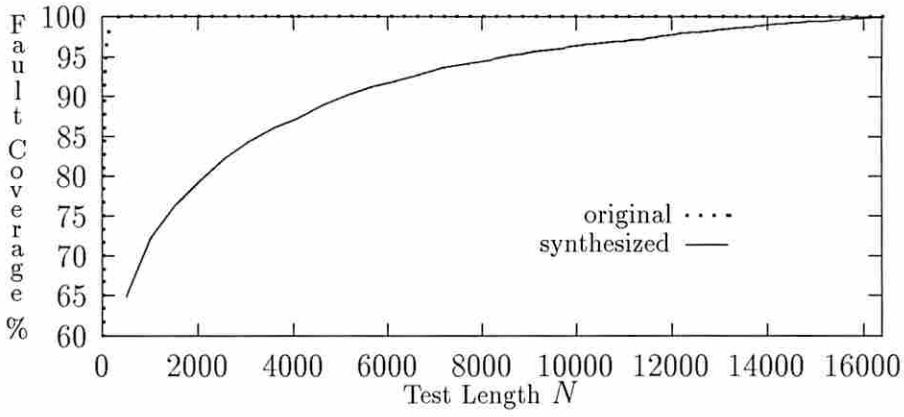
Some PLA benchmark circuits [4] were synthesized using testable synthesis procedures [14] incorporated in *SIS* (see [5]). These circuits were also designed by hand from their truth table descriptions. On the other hand, truth tables of ALU 74181 (from the

TTL handbook and called c181 in the following) and c432 and c880 (from the ISCAS 85 benchmark set [6]) were derived from their circuit descriptions. (In fact the truth tables of c432 and c880 could not be derived due to the large size of the truth tables. Hence, truth tables were derived for partitioned versions [18] of these circuits.) These truth tables were input to the logic synthesis procedure and synthesized versions of these circuits were obtained. LFSR generated pseudo-random patterns were then used to compare test-length vs. fault coverage relationship for hand designed and automatically synthesized circuits for each function. The data was averaged over several pseudo-random sequences generated using different LFSR feedback polynomials and seeds. It was found that for each function, the random pattern test length required to obtain high fault coverage was significantly higher for the synthesized the circuit. Figures 1 (a)-(c) show typical comparisons.

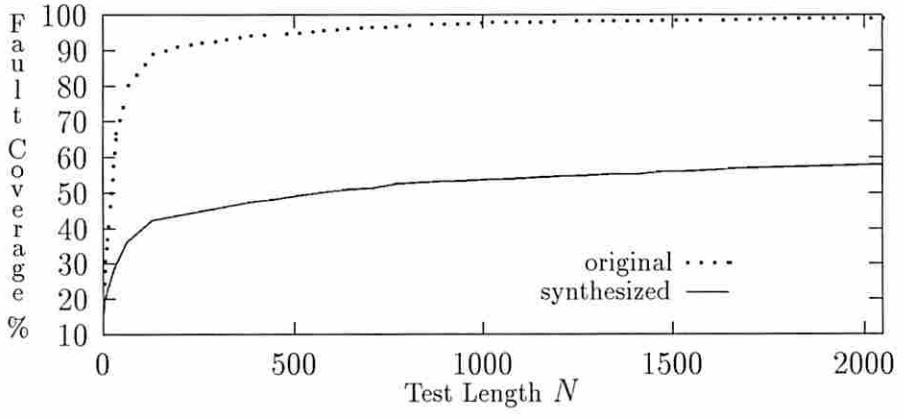
These results indicate that circuits synthesized by typical synthesis procedures are not random pattern testable. Existence of other implementations of the same boolean functions shows that the presence of random pattern resistant faults is not due to the nature of these functions but is a consequence of the circuit structure generated by the synthesis procedure. Also, a significant difference in the random pattern testabilities between hand designs and synthesized circuits demonstrates that successful development of random-pattern testable logic synthesis procedures will be very useful.

2.2 Review

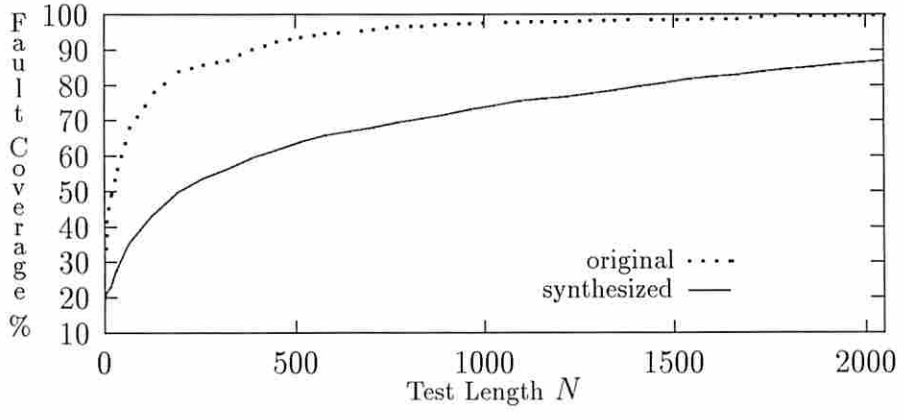
Early research in area of testable logic concentrated on the identification and elimination of redundancies from combinational circuits [3]. It has been shown that all single stuck-at faults in any prime and irredundant two-level realization of a circuit are fully testable. Procedure for synthesis of circuits in which all single and multiple stuck-at faults are testable has been reported in [7]. In recent years, testability preserving logic synthesis transformations have been studied. It is shown in [10] that algebraic transformations preserve testability of multiple stuck-at faults. Hence, if all multiple stuck-at faults in a given circuit are testable, then all multiple stuck-at faults in a circuit obtained by application of algebraic



(a) ALU 181



(b) Partitioned c432



(c) Partitioned c880

Figure 1: Comparisons of fault coverage obtained by pseudo-random test sequence.

transformations to the circuit are also testable. It has been demonstrated in [11, 12] that careful assignment of don't cares of functions can improve the random pattern testability of the resulting circuit. (However in some cases, their examples have undetectable single stuck-at faults.) In [14], a logic synthesis procedure based on concurrent extraction of single and double cube divisors and their complements has been presented. The procedure has been proven to preserve the testability of single stuck-at faults. Hence, if all single stuck-at faults in the circuit input to the synthesis procedure are testable, then all single stuck-at faults in the circuit generated by the procedure are also testable. (This procedure is available as *fast_extract* (*fx*) within *SIS*.) Finally, a logic synthesis procedure to reduce deterministic test length has been presented in [8].

As illustrated above, two realizations of the same boolean functions (all stuck-at faults testable in both) can differ widely in their random pattern testabilities (see Figure 1 (a)-(c)). The objective of this research is to develop a logic synthesis procedure that enhances random pattern testability of the circuit while preserving testability of single stuck-at faults. An approach similar to ours (using algebraic transformations) has been independently developed [19].

2.3 Definitions and Notations

In the following, some definitions are presented. The reader is referred to [5, 14] for the formal definitions.

Consider an n-input, m-output combinational circuit. Let $X = \{x_1, x_2, \ldots, x_n\}$ be n input variables. Let $F = \{f_1, f_2, \ldots, f_m\}$ be m combinational boolean functions of the input variables. Let $Y = \{y_1, y_2, \ldots, y_m\}$ be the circuit outputs that implement these boolean functions. A literal is either an input or intermediate variable z_i or its compliment $\overline{z_i}$. A cube is a product (AND) of one or more literals. Each variable appears at most once in a cube either in complemented or uncomplemented form. An expression is a sum (OR) of one or more cubes. An expression E is said to be *cube-free* if the only cube that can divide E is 1. An expression g is called an algebraic divisor of expression f if $f = gq + r$ where q and

r are expressions and $q \neq 0$. An algebraic divisor g of f is called a *kernel* if g is cube-free. If divisor g is a single cube, then g is called a single cube divisor. If divisor g is a sum of two cubes, then g is said to be a double-cube divisor.

Subsets of double-cube divisors are represented by $D_{x,y,s}$, where x ($\leq y$) is the number of literals in the first cube, y is the number of literals in the second cube and s is the total number of distinct variables that appear in the double cube. Subsets of single-cube divisors are denoted by S_k, where k is the number of literals in the single-cube divisor. In the following, notion of detectability profile of a circuit is presented.

2.4 Detectability Profile

Random pattern testability is itself difficult to quantify in a manner that can be incorporated into logic synthesis procedures. However, the detectability profile can be used to guide logic synthesis. Detectability profile has been used to estimate random pattern test sequence length required to attain desired fault coverage for a given circuit [2, 13, 16, 20]. Given an n-input circuit, the number of tests for any fault in the circuit can vary between 0 (undetectable) and 2^n. Let h_k be the number of faults in a circuit which have exactly k tests. Then

$$H = \{h_0, h_1, \ldots, h_{2^n}\}$$

is called the detectability profile of the circuit. Many results have been derived relating the detectability profile H and the random pattern test length required to attain a desired fault coverage. In the following, some simple empirical observations [13, 16, 20] will be presented. They are useful because complex relationship between the detectability profile and random test sequence length can not be easily used to direct the logic synthesis.

1. The faults with few tests (k_{min}) are the hardest to detect. Empirically, the coverage of faults with $k_{min}, k_{min+1}, \ldots, 2 * k_{min}$ tests (called hard-to-detect faults) requires long test sequences while the other faults are detected by much shorter sequences. Hence, logic synthesis procedure should improve the detectability of these faults.

2. Among the hard to detect faults, lower the number of tests for a given fault, higher is its impact on the random pattern test length. Hence, maximization of k_{min} is desirable.

3. If k_{min} cannot be increased, then $h_{k_{min}}$ should be minimized. If that cannot be achieved, then $h_{k_{min}+1}$ should be minimized, and so on.

These observations are used in the following to define a cost function that directs the proposed synthesis procedure.

3 Random Pattern Testable Logic Synthesis

One of the most important steps in automatic synthesis of multilevel logic is the identification of subexpressions/subfunctions that are used several times in the given circuit. Identification of these subexpressions permits circuit simplification because a common subexpression/subfunction can be implemented only once and its output can fan-out to several parts of the circuit. This helps reduce the circuit area. This process is implemented as decomposition and/or factorization. Cubes, kernels and double-cubes are the classes of subexpressions most commonly targeted by many logic synthesis procedures. In traditional logic synthesis procedures, subexpressions are selected for decomposition based on the circuit area (measured by the number of literals) that can be saved by their extraction. In many cases, the resulting circuit may have undetectable stuck-at faults.

Very simplified multilevel logic synthesis procedure can be described using three steps. First, a set of possible divisors is computed (cubes, kernels, double-cubes, etc.). Next, a cost which reflects the merit of selecting the given divisor for extraction is assigned to each divisor. Third, a candidate for extraction is selected based on its cost. After each extraction (in some cases after a number of extractions), the set of possible divisors and their costs are recomputed and the procedure is repeated.

The above describes the basic outline of the proposed procedure as well. The candidates are single cube divisors with two literals and double cube divisors used in *fx* [15]. This decision was based on three main considerations. Firstly, a procedure based on these

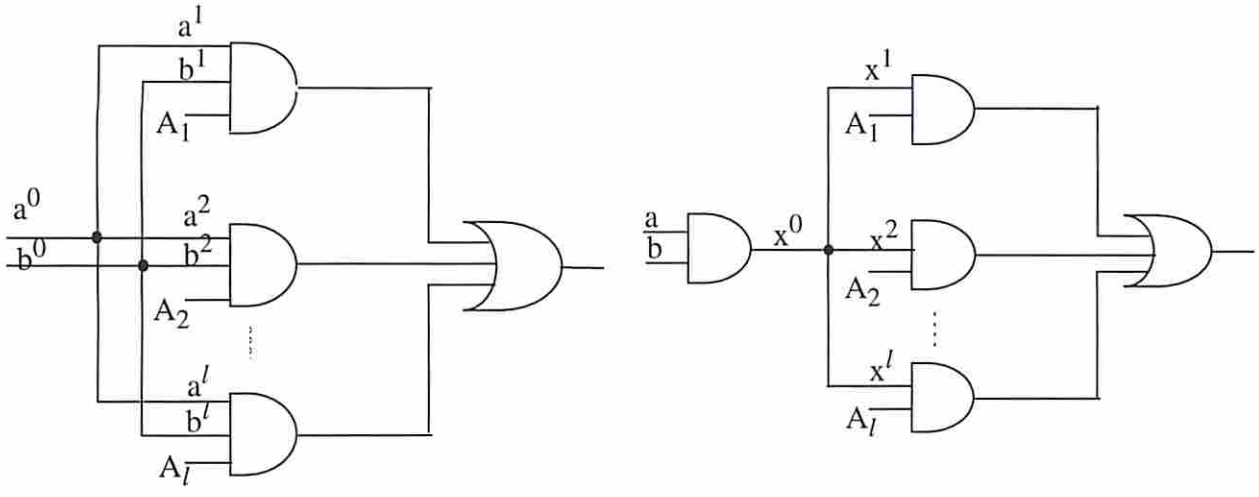
candidates is guaranteed to preserve testability of all single stuck-at faults [15]. This guarantees that the proposed procedure synthesizes circuits with no undetectable single stuck-at faults, provided all single stuck-at faults in the input circuits are testable. Secondly, as shall be seen in the following, the simplicity of these candidates allows comprehensive analysis of the impact of extraction of each divisor on the detectabilities of various faults in the modified circuit. Lastly, the number of candidates (objects of size two) is always in polynomial domain. Also great cost reduction, in terms of the number of literals, have been obtained by their use in traditional logic synthesis.

However, since the objective of the proposed algorithm is to generate logic that is random pattern testable, the cost assignment to the various single and double cubes is different from the cost that is used in *fx* (number of literals saved). In this section, we will concentrate on the description of cost that reflects improvement in random pattern testability resulting from the extraction of each single/double cube divisor.

Hard to Detect Faults: As was discussed earlier, the stuck-at faults with the fewest test vectors have the most adverse impact on the length of random pattern test sequence required to test the circuit. Hence, in the following, only the impact of various extractions on the hard to detect faults will be considered. Hence, if a particular divisor extraction improves only the detectability of faults that already have many tests, then the divisor is assigned a lower cost.

Let k_{min} be the number of tests for the stuck-at fault with the fewest test vectors. (Recall that h_k is the number of faults that have k tests.) Hence, $h_0 = h_1 = \ldots = h_{k_{min}-1} = 0$ and $h_{k_{min}} \neq 0$. Let $k_2 > k_{min}$ be such that $h_{k_2} \neq 0$ and $h_{k_{min}+1} = \ldots = h_{k_2-1} = 0$. In this work, all faults that have $k_{min}, k_2, \ldots, k_p$ tests are classified as hard to detect where $k_p = const * (k_2 - k_{min})$. The impact of various extractions will be studied only with respect to the detectabilities of the faults in this set of hard to detect faults ($HTDF$).

Impact of Extraction on Detectabilities: Any extraction changes the structure of the circuit. A change in the structure of the circuit changes the test sets for various faults. Also,



(a) single cube divisor ab (b) circuit after extraction

Figure 2: Extraction of a single cube divisor ab.

some lines in the circuit are eliminated while new lines are created. To compute the cost of a single/double cube divisor, it is essential to understand the exact impact of the extraction on the detectabilities of the (hard-to-detect) faults. In the following, the impact of single cube extraction and double cube extraction (from single output) on testability of various faults will be discussed to illustrate the computation of the cost.

Consider the subcircuit shown in Figure 2 (a). Let ab be the single cube divisor being considered for extraction. A_i $(1 \leq i \leq l)$ are cubes of primary inputs and/or intermediate variables. Let $T(x_s)$ denote the test set for the fault line-x stuck-at-s, where $s \in \{0, 1\}$. Figure 2 (b) shows the subcircuit after the extraction of the cube ab. The cube extraction eliminates lines $a^1, a^2, \ldots, a^l$ and $b^1, b^2, \ldots, b^l$. Instead, a and b are inputs to the AND gate implementing the extracted cube $x^0 = ab$. The output x^0 fans out $x^1, x^2, \ldots, x^l$ to the AND gates (which now have one less input than in the original circuit).

Consider the test sets of the faults in the circuit shown in Figure 2 (b). They can be

expressed in terms of the test sets for the faults in the original circuit (Figure 2 (a)) as:

$$T(x_0^i) = T(a_0^i) = T(b_0^i)\,, \forall 0 \leq i \leq l. \tag{1}$$

Note that the concepts of fault equivalence and dominance in [1] have been used to derive all these results which are stated here without proof (see [9] for details). Hence, the detectability of the stuck-at-0 faults on lines x^i is the same as the detectability of the stuck-at-0 faults on lines $a^i(b^i)$ in the original circuit, for all $0 \leq i \leq l$. On the other hand, the stuck-at-1 fault in the newly added lines x^i has a larger test set than either a^i or b^i. It can be also shown that

$$T(x_1^i) \supset T(a_1^i)\,, \forall 0 \leq i \leq l,$$

and

$$T(x_1^i) \supset T(b_1^i)\,, \forall 0 \leq i \leq l.$$

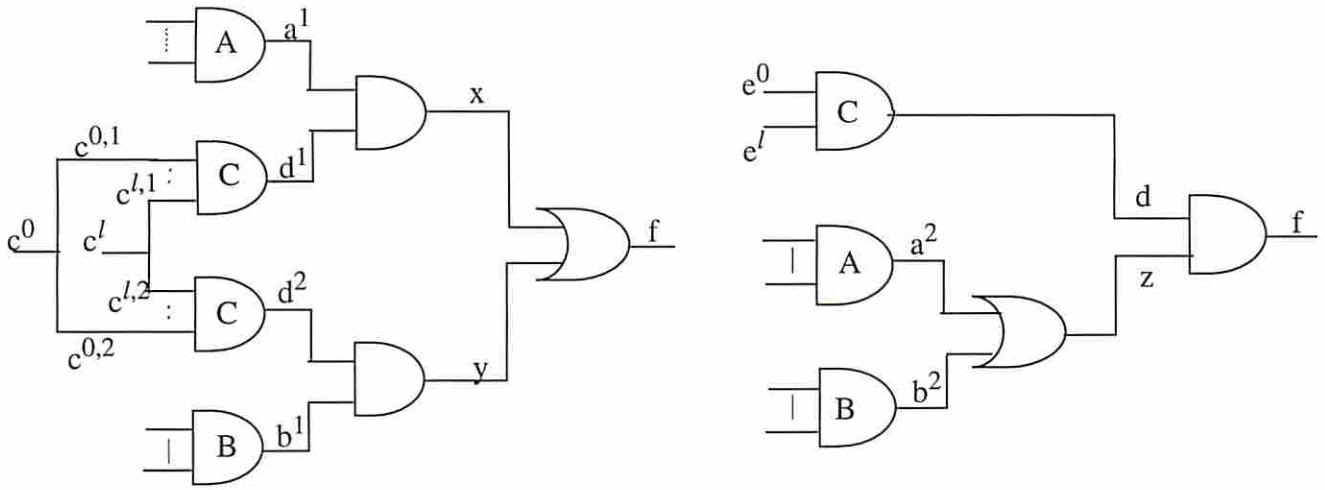
Hence the test set of x^i stuck-at-1 is given by

$$T(x_1^i) \supset T(a_1^i) \cup T(b_1^i)\,, \forall 0 \leq i \leq l. \tag{2}$$

This implies that the stuck-at-1 faults at the newly created lines x^i $(0 \leq i \leq l)$ are easier to detect than stuck-at-1 faults in the lines a^i (and b^i) $(0 \leq i \leq l)$ in the original circuit. Furthermore, since $T(a_1^i) \cap T(b_1^i) = \phi$, $|T(x_1^i)| \geq |T(a_1^i)| + |T(b_1^i)|$ for all $0 \leq i \leq l$. The test sets of the remaining faults in this subcircuit do not change.

Let $S_{single\_cube(c)}$ denote the set of s-a-0/1 faults whose detectabilities change due to the extraction of single cube $c = ab$. Note that a_0^i and b_0^i are replaced by x_0^i which has the same testability. However, two lines are replaced by one, thereby impacting the detectability profile. Also, a_1^i and b_1^i are replaced by x_1^i which is easier to detect. Hence, $S_{single\_cube(c)} = \{a_s^i, b_s^i | 1 \leq i \leq l, s = 0,1\}$ is the set of faults whose detectabilities are improved in one way or other by the extraction of single cube c. The faults that should be considered for the evaluation of the cost of the single cube extraction are given by $S_{single\_cube(c)} \cap HTDF$. The cost associated with the extraction of single cube c is hence given by:

$$cost(c) = \sum_{f \in S_{single\_cube(c)} \cap HTDF} weight(f),$$



(a) double cube divisor $A + B$ (b) circuit after extraction

Figure 3: Extraction of a double cube divisor $A + B$ in a single output.

where f is any single stuck-at fault and

$$weight(f) = k_p - |T(f)|.$$

Now, consider the subcircuit shown in Figure 3 (a). Let $A + B$ be the double cube divisor being considered for extraction in the single output. Figure 3 (b) shows the subcircuit after the extraction of the divisor $A+B$. The extraction of $A+B$ eliminates lines d^1, d^2, x, y and $c^{0,1}, c^{0,2}, \ldots, c^{l,1}, c^{l,2}$.

However, as the structure of this subcircuit changes, so do the testabilities of faults in lines a^2, b^2, d, f, z and $e^0, e^1, \ldots, e^l$. Consider the test sets of these lines in the circuit shown in Figure 3 (b). For stuck-at-0 faults, they can be expressed in terms of the test sets for the faults in the original circuit (Figure 3 (a)) as:

$$T(a_0^2) = T(a_0^1). \tag{3}$$

$$T(b_0^2) = T(b_0^1). \tag{4}$$

$$T(e_0^i) = T(d_0) = T(f_0) = T(z_0) \supset T(a_0^1) \cup T(b_0^1)\,, \forall 0 \leq i \leq l. \tag{5}$$

On the other hand, for the stuck-at-1 faults of these lines, it can be also shown that:

$$T(z_1) = T(a_1^2) = T(b_1^2) = T(a_1^1) = T(b_1^1). \tag{6}$$

$$T(e_1^i) = T(c_1^i)\,, \forall 0 \leq i \leq l. \tag{7}$$

$$T(d_1) \supset \bigcup_{0 \leq i \leq l} T(e_1^i) = \bigcup_{0 \leq i \leq l} T(c_1^i). \tag{8}$$

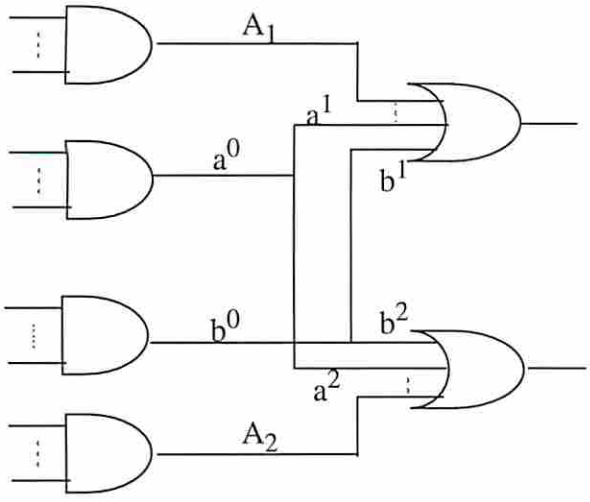
$$T(f_1) \supset T(z_1) \cup T(d_1) \supset T(a_1^2) \cup \bigcup_{0 \leq i \leq l} T(c_1^i). \tag{9}$$

These equations imply that the stuck-at faults in the newly generated lines (due to the change of the circuit structure) are equivalent or easier to detect than stuck-at faults in the original circuit. Hence, $S_{double\_cube\_single\_output(A+B)} = \{d_s^1, d_s^2, x_s, y_s, c_s^{i,1}, c_s^{i,2} | 0 \leq i \leq l, s = 0, 1\}$ can be used to compute the cost function for extraction of $A + B$ from Figure 3 (a).

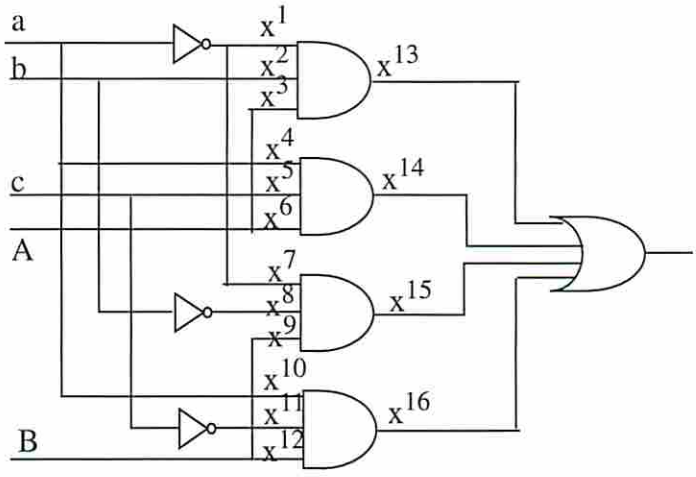
Similarly cost functions for double-cube divisor in multiple outputs, $D_{2,2,2}$ and $D_{2,2,3}$ can be computed. The faults whose detectabilities can be improved by these extractions are shown in Figure 4 (a)-(c).

The above describes the proposed procedure and the method to compute the costs for various single/double cube extractions. Computation (and recomputation) of single and double cube divisors can be carried out using the procedure used by *fx*. However, computation of $T(f)$ at the beginning and its recomputation after each extraction needs new procedures.

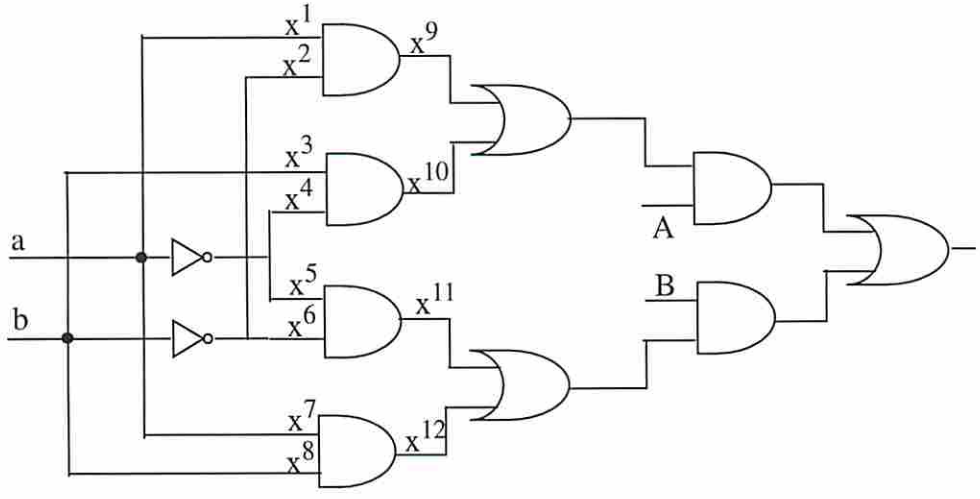
Computation of $T(f)$: The input to the proposed multilevel logic synthesis is a two-level fully testable *AND-OR* circuit. For two level circuits, $T(f)$ can be computed for all faults using the procedure outlined in [9]. In practice, only $|T(f)|$ is computed for all faults. The procedure can be easily applied to most PLA benchmark circuits. However, after each extraction, the circuit structure and the detectabilities of some faults change. Cost must be recomputed for various candidates (the set of candidates is also recomputed). Computation



(a) $S_{double\_cube\_multiple\_outputs(D_{1,1,2})} = \{a_s^1, a_s^2, b_s^1, b_s^2 | s = 0, 1\}$



(b) $S_{D_{2,2,3}} = \{x_0^i, x_1^j | 1 \leq i \leq 16, 1 \leq j \leq 12\}$



(c) $S_{D_{2,2,2}} = \{x_s^i | 1 \leq i \leq 12, s = 0, 1\}$

Figure 4: Extraction of (a) double-cube divisor in multiple outputs, (b) $D_{2,2,3}$ and (c) $D_{2,2,2}$.

of exact $|T(f)|$ value for general multilevel circuit is difficult. However, the values of $|T(f)|$ can be recomputed approximately using the results shown above. Consider the case of single cube extraction. Equations 1 and 2 can be used to compute lower bounds on the values of $|T(x_0^i)|$ and $|T(x_1^i)|$ $(0 \leq i \leq l)$. Equations 3-9 can be used to recompute lower bound on the test set sizes for the faults in the lines created by the extraction of double cube divisor $A + B$. Similar equations have been derived for other divisors and can be used to compute lower bounds on $|T(f)|$ after each extraction. Note that after the first few extractions, $|T(f)|$ values are inaccurate (but they are lower bounds on the actual detectabilities) and may influence future extractions adversely. Procedures to accurately recompute $|T(f)|$ are being developed.

4 Experimental Results

The proposed procedure for random pattern testability directed single and double cube divisor extraction has been implemented as *tstfx* within *SIS 1.0*. As outlined above, the procedure is analogous to *fx*. The proposed cost function replaces literal savings as the divisor selection criterion. Algorithms to compute test set size for all stuck-at faults have been implemented. Also, procedures to update test sets for stuck-at faults after each extraction have been implemented. In the following experiments, the input to both *tstfx* and *fx* is a two-level prime and irredundant implementation of the given function.

Before comparing *tstfx* and *fx*, we present random pattern fault coverage for some circuits synthesized using *algebraic*, *boolean* and *rugged* scripts of *SIS*. Table 1 shows the comparison when input to these scripts is a two level implementation obtained by applying *resub -a; simplify -d* on the benchmark PLA circuits. These scripts were repeatedly applied until no literal count reduction occurred between consecutive iterations. The random testability of the resulting circuits are obtained by using LFSR generated pseudo-random sequences. Data presented is average over several different sequences obtained by LFSR with different feedback polynomials and seeds (typically 5-10 different pseudo-random sequences are used). Table 2 shows similar comparison when the input to the scripts is a two level

Table 1: Comparison of *algebraic, boolean,* and *rugged* scripts.

CKT	*Area*							*Testability*			
	I/O	*Lit(fac)*			*Levels*			*Test*	*FCOV%*		
		alge	*bool*	*rugg*	*alge*	*bool*	*rugg*	*Length*	*alge*	*bool*	*rugg*
in2	19/10	497	490	435	10	8	10	2^{19}	98.04	98.94	99.80
in7	26/10	256	137	114	8	7	6	2^{18}	83.33	96.33	99.43
vg2	25/8	227	126	88	7	5	5	2^{20}	96.36	98.91	217088*
x1dn	27/6	216	105	89	8	5	6	2^{20}	96.56	98.74	137216*

* indicates average test length when 100% stuck-at fault coverage was achieved.

Table 2: Comparison of *algebraic, boolean,* and *rugged* scripts with optimized two-level circuit as input.

CKT	*Area*							*Testability*			
	I/O	*Lit(fac)*			*Levels*			*Test*	*FCOV%*		
		alge	*bool*	*rugg*	*alge*	*bool*	*rugg*	*Length*	*alge*	*bool*	*rugg*
in2	19/10	533	491	319	9	7	8	2^{19}	99.02	99.70	99.87
in7	26/10	134	113	117	7	7	6	2^{18}	99.93	99.68	99.84
vg2	25/8	193	95	88	6	5	5	2^{20}	93.08	99.53	217088*
x1dn	27/6	211	104	90	7	5	6	2^{20}	93.93	99.15	137216*

* indicates average test length when 100% stuck-at fault coverage was achieved.

circuit obtained by using *espresso* (with *single_output* option) to obtain optimized two-level circuits.

The comparison of literal counts is consistent in both experiments. The resulting circuit size is typically smaller for *rugged* script than *boolean* script which in turn is better than *algebraic* script. The random testabilities for both experiments follow the same trend as literal counts. However, the most important observation is that for circuit *in*2, none of the standard scripts produced an irredundant multilevel circuit. This is observed for many other circuits as well. Comparison of Table 1 and Table 2 shows that starting with a prime and irredundant cover in each output produced better quality multilevel circuits in terms of both area and random testability, even though some circuits still had undetectable faults.

Now the data comparing proposed *tstfx* and existing *fx* will be presented. Table 3 compares the characteristics of the resulting circuits obtained by the application of these two procedures. (For better measure the merit of *fx*, we ran *simplify -m nocomp; resub -a; fx; eliminate 0* [17] on benchmark PLA circuits.) Surprisingly, the circuits obtained by application of *tstfx* have fewer literals in most cases. Table 4 compares the random-pattern testability of circuits obtained by using *tstfx* and *fx*. The random pattern test length vs. fault coverage data is averaged over several different LFSRs and seeds as outlined above. Since some of the circuits have large number of inputs, pseudo random sequences with lengths only a fraction of exhaustive test length could be used due to the large number of simulations necessary. The data clearly shows that a large decrease in random pattern test length results for 98% and 99% fault coverage. The reduction in test length varied from 4% to 90%. In most cases, the random pattern test length was decreased by half. Considering the fact that this was accomplished with a simultaneous decrease in circuit area shows that this approach to random pattern testable logic synthesis is very effective.

Figure 5 compares the random pattern test length vs. fault coverage characteristics for implementations of ALU181 obtained by using *tstfx* and *fx*. This shows a typical trend observed for most circuits. Figure 6 incorporates the random pattern testability of circuits obtained by the proposed *tstfx* into Figure 1 (a). This clearly shows both the improvement obtained by the proposed *tstfx* and at the same time further improvements that may be possible.

The results of *tstfx* are very encouraging. They clearly show that random-pattern testability can be enhanced by using the proposed methodology. However, further development is necessary to enable the application of additional simplification steps to further optimize the resulting circuit. For example, in *rugged* script, the circuit obtained by *fx* is simplified and *fx* is reapplied. This leads to much smaller circuits. However in many cases, the simplification generates circuits with undetectable stuck-at faults. In some cases, circuits synthesized by *rugged* script require shorter random test sequences when compared to the circuit implemented using *tstfx* (for example, for circuit *in5* the average random pattern test length required for the circuit designed using *rugged* is half of that for the circuit designed

Table 3: Comparison of circuits obtained by using *tstfx* and *fx*.

CKT	*I/O*	*Nodes*		*Lit(fac)*			*Levels*	
		sisfx	*tstfx*	*sisfx*	*tstfx*	**%imp**	*sisfx*	*tstfx*
b10	15/11	197	121	670	554	17.31	10	11
b4	33/23	109	95	365	395	-8.22	8	13
c181	14/8	781	332	2533	1519	40.03	8	15
chkn	29/7	200	99	699	509	27.18	9	15
gary	15/11	220	126	728	612	15.93	11	11
in2	19/10	265	142	861	556	35.42	9	12
in4	32/20	370	190	1132	711	37.19	11	15
in5	24/14	133	107	442	373	15.61	10	13
in6	33/23	109	95	366	405	-10.65	8	13
in7	26/10	80	60	242	174	28.10	9	13
rckl	32/7	132	102	430	329	23.49	10	12
vg2	25/8	139	67	461	283	38.62	7	14
x1dn	27/6	148	148	440	449	-2.04	8	17

Table 4: Random-pattern testability of circuits obtained by using *tstfx* and *fx*.

CKT	k_{min} for 2 level input	*Test Length*						*Final Data*		
		At 98% Fcov			*At 99% Fcov*			*Test Length*	*FCOV%*	
		sisfx	*tstfx*	**%imp**	*sisfx*	*tstfx*	**%imp**		*sisfx*	*tstfx*
b10	2	10240	6656	35.00	11264	10240	9.09	2^{15}	22912*	22912*
b4	262144	40960	25088	38.75	71552	40960	42.75	-	122176*	122176*
c181	1	12160	7168	41.05	13760	9470	31.18	2^{14}	16288*	16096*
chkn	32	-	-	-	-	-	-	2^{22}	95.30	97.60
gary	2	10240	9216	10.00	11776	11264	4.35	-	22912*	22912*
in2	64	7680	4608	40.00	10752	5632	47.62	2^{19}	31744*	17920*
in4	4096	302592	63488	79.02	686080	111616	83.73	2^{20}	99.57	99.96
in5	1024	13824	5632	59.26	17920	8192	54.28	2^{18}	99.73	38912
in6	262144	40960	25088	38.75	71552	40960	42.75	-	122176*	122176*
in7	512	123392	14336	88.37	205632	20480	90.04	2^{18}	99.11	99.94
rckl	1	-	-	-	-	-	-	2^{22}	79.07	80.62
vg2	64	457216	217088	52.52	605696	263168	56.55	2^{20}	99.83	888576*
x1dn	96	-	552320	-	-	743552	-	2^{20}	97.85	99.87

* indicates average test length when 100% stuck-at fault coverage was achieved.

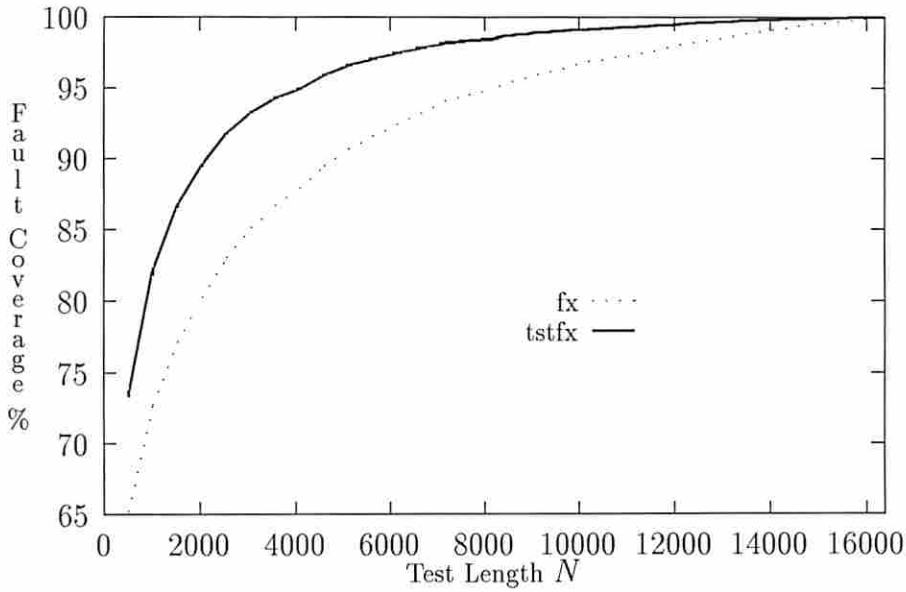


Figure 5: Random pattern fault coverages of ALU 181 synthesized by *tstfx* and *fx*.

by *tstfx* only). However, typically, the circuits designed using *tstfx* are more random pattern testable than those obtained by *rugged* script. In many cases, circuits synthesized with *rugged* script even have undetectable faults. The results show that simplification procedures that do not explicitly consider random pattern testability can result in circuits that are not random-pattern testable in general. Hence there is a need to develop further simplification procedures that preserve/enhance random pattern testability. Such procedures are under consideration.

5 Conclusion

A logic synthesis procedure *tstfx* that synthesizes circuits that are random pattern testable has been presented. The procedure is analogous to *fast_extract* (*fx*) in *SIS*. The procedure is based on the analysis of the improvement of the detectability of various faults due to any extraction. The improvement in the detectability of hard-to-detect faults due to the extraction of any divisors is used to direct the multilevel logic synthesis procedure. Experimental

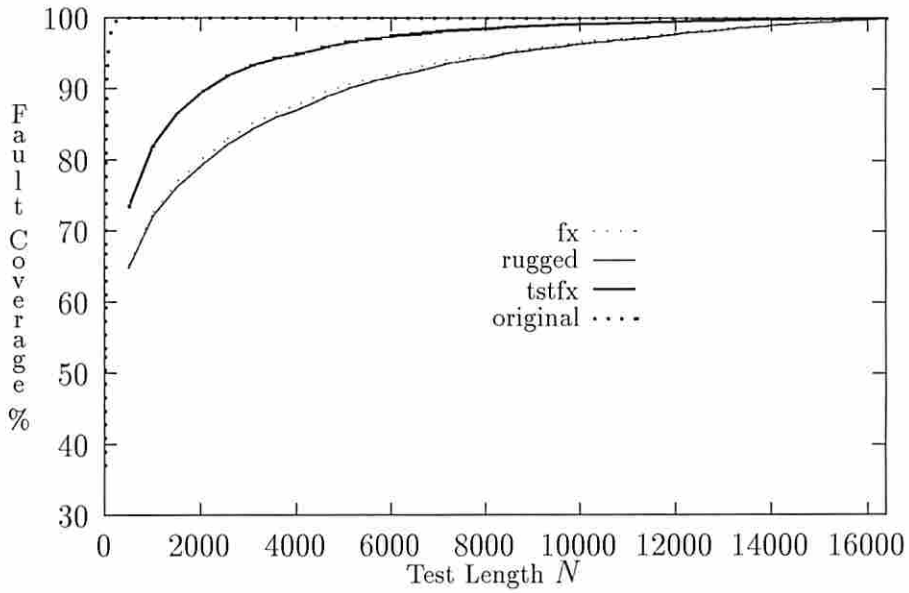


Figure 6: Random pattern fault coverages of ALU 181 synthesized by *tstfx, fx, rugged* and original $TTL74181$.

results indicate that the random pattern testability of circuits synthesized by the proposed *tstfx* are significantly higher than those synthesized by *fx*. In most examples the decrease in random pattern test length is accompanied by reduction in circuit size. Hence, the proposed logic synthesis procedure can greatly simplify BIST hardware design and/or improve fault coverage.

Currently, procedures to enhance the quality of the detectability estimates ($|T(f)|$) for faults in the circuit are being developed. Simplification procedures that preserve/enhance random pattern testability of the resulting circuits are being studied. They will enable iterative application of *tstfx* to further reduce circuit size and enhance its random pattern testability.

References

[1] M. Abramovici, M. A. Breuer, and A. D. Friedman. *Digital Systems Testing and Testable*

Design. Computer Science Press, New York, N.Y., 1990.

[2] P. H. Bardell, W. H. McAnney, and J. Savir. *Built-In Test for VLSI: Pseudorandom Techniques.* John Wiley & Sons, 1987.

[3] D. Brand. Redundancy and Don't Cares in Logic Synthesis. *IEEE Trans. on Computer*, C-32(10):947–952, October 1983.

[4] R. K. Brayton, G. D. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli. *Logic Minimization Algorithms for VLSI Synthesis.* Kluwer Academic Publishers, Boston, MA, 1984.

[5] R. K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. R. Wang. MIS: A Mutiple-Level Logic Optimization System. *IEEE Trans. on CAD*, CAD-6(6):1063–1080, November 1987.

[6] F. Brglez and H. Fujiwara. Neutral Netlist of Ten Combinational Benchmark Circuits and a Target Translator in FORTRAN. In *Proceedings IEEE International Symposium on Circuits and Systems*, 1985.

[7] R. Dandapani and S. Reddy. On the Design of Logic Networks with Redundancy and Testability Considerations. *IEEE Trans. on Computer*, C-23(11):1139–1149, November 1974.

[8] K. De and P. Banerjee. Logic Partition and Resynthesis for Testability. In *Proceedings IEEE International Test Conference*, pages 906–915, 1991.

[9] S. K. Gupta and C.-H. Chiang. Random Pattern Testable Logic Synthesis. Technical Report CENG 94-08, University of Southern California, 1994.

[10] G. Hachtel, R. Jacoby, K. Keutzer, and C. Morrison. On Properties of Algebraic Transformations and the Multifault Testability of Multilevel Logic. In *Proceedings IEEE International Conference on Computer-Aided Design*, pages 422–425, 1989.

[11] A. Krasniewski. Can Redundancy Enhance Testability. In *Proceedings IEEE International Test Conference*, pages 483–491, 1991.

[12] A. Krasniewski and A. Albicki. Random Testability of Redundant Circuit. In *Proceedings IEEE International Conference on Computer Design*, pages 424–427, 1991.

[13] A. Majumdar and S. Sastry. On the Distribution of Fault Coverage and Test Length in Random Testing of Combinational Circuit. In *Proceedings IEEE-ACM Design Automation Conference*, pages 341–346, 1992.

[14] J. Rajski and J. Vasudevamurthy. Testability Preserving Transformation in Multi-level Logic Synthesis. In *Proceedings IEEE International Test Conference*, pages 265–273, 1990.

[15] J. Rajski and J. Vasudevamurthy. The Testability-Preserving Concurrent Decomposition and Factorization of Boolean Expressions. *IEEE Trans. on CAD*, 11(6):778–793, June 1992.

[16] J. Savir and P. H. Bardell. On Random Pattern Test Length. *IEEE Trans. on Computer*, C-33(6):467–474, June 1984.

[17] H. Savoj. *Don't Cares in Multi-Level Network Optimization.* PhD thesis, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, 1992.

[18] R. Srinivasan, S. K. Gupta, and M. A. Breuer. An Efficient Partitioning Strategy for Pseudo-Exhaustive Testing. In *Proceedings IEEE-ACM Design Automation Conference*, pages 242–248, 1993.

[19] N. A. Touba and E. J. McCluskey. Automated Logic Synthesis of Random Pattern Testable Circuits. *Submitted to IEEE International Test Conference*, 1994.

[20] K. D. Wagner, C. K. Chin, and E. J. McCluskey. Pseudorandom Testing. *IEEE Trans. on Computer*, C-36(3):332–343, March 1987.