Fault Coverage Estimation for Non-Random Functional Input Sequences

Soumitra Bose
Intel Corporation
Folsom, CA 95630
soumitra.bose@intel.com

Vishwani D. Agrawal
Auburn University
Auburn, AL 36849
vagrawal@eng.auburn.edu

Abstract

Statistical stuck-at fault coverage estimation assumes that signals at primary inputs and at other internal gates of the circuit are statistically independent. While valid for random and pseudo-random inputs, this causes substantial errors in coverage estimation for input sequences that are functional and not random, as shown by experimental data presented in this paper. At internal gates, signal correlation due to fanout reconvergence, even for random input sequences, contributes to errors. A significantly improved coverage estimation algorithm is presented in this paper. First, during logic simulation we identify faults that are guaranteed to stay undetected by the applied vectors. Then, after logic simulation, we estimate the detection probabilities of the remaining faults. Compared to Stafan, the statistics gathered during logic simulation are modified in order to eliminate the non-random biasing of the input sequence. Besides the improved detection probabilities, a newly defined effective length (N_{eff}) of the vector sequence corrects for the temporally correlated signals. Experimental results for ISCAS combinational benchmarks demonstrate validity of this approach.

1 Introduction

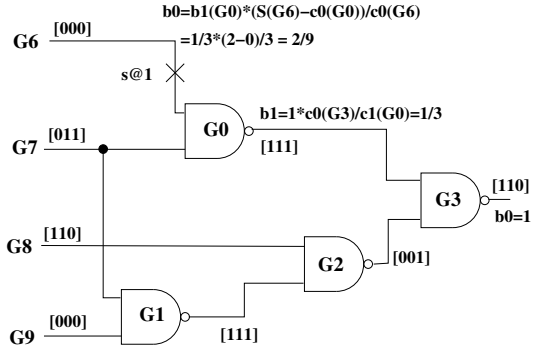
Simulation [5, 16] of stuck-at faults is known to be a resource intensive task and several acceleration techniques have been proposed [4, 7, 9, 8, 14]. Verification of industrial designs often involves simulation of large number of functional vectors that can be several million cycles long. Typically, a million gate design with a 2% fault sample may require several days to simulate a long input sequence. Moreover, numerous such functional sequences have to be simulated. It is estimated that a modern microprocessor design requires about a month to complete fault grading, a step that needs to be repeated when relatively minor changes are made during the design cycle.

Statistical fault coverage estimation [10], proposed as an alternative to fault simulation, suffers from a lack of accuracy when input vectors are not random. Such estimation techniques involve logic simulation with a low overhead for collection of relevant statistical data and a subsequent post-processing step that evaluates the detection probability of each fault. Another approximate method for coverage estimation is critical path tracing [1]. While critical path tracing can be more accurate than statistical methods, its overhead over logic simulation is substantially higher. A circuit traversal is required after simulation of each clock cycle, making it impractical for long input sequences.

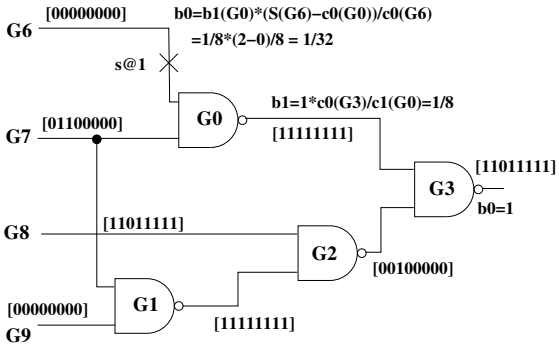
We propose a two step approach that first involves monitoring of signal conditions to help identify faults that remain undetected, followed by a post-processing step to evaluate the detection probabilities of the remaining faults. The first step is common to another algorithm, referred to as upper-bounding of fault coverage, that has been reported in a recent paper [2]. While the upper-bounding procedure finds faults that are guaranteed to remain undetected by a sequence, the second phase of the algorithm reported in this paper eliminates faults that have low detection probabilities. The upper-bounding algorithm has no faults that are incorrectly estimated to remain undetected (hence the name "upper-bounding"). The algorithm in this paper generates a detection probability for each fault and incurs false negative errors when faults with low detection probabilities are removed from the detection list. The number of false positive errors is also lower, resulting in fault coverage estimates that are lower than the upper bound, and hence closer to those obtained from exact fault simulation. The detection probabilities that are generated are somewhat similar to Stafan [10]. Stafan assumes the input sequence to be random, and is shown to cause errors for sequences that are biased due to temporal correlations. Stafan's statistical data collection during logic simulation is modified to remove such biasing.

Along with relevant background on Stafan, Section 2 presents experimental data that demonstrate the effect of signal correlation for biased input streams on fault coverage estimation. Section 3 presents the algorithm used for choosing signal conditions that are monitored, as illustrated with an example. Section 4 presents the algorithm for evaluation of fault detection probabilities. Section 5 illustrates the algorithm of Section 4 with the example of Section 3. Section 6 illustrates the difference between the upper-bounding [2] and the work presented in this paper. Section 7 presents experimental results for ISCAS circuits, and demonstrates the speedup and ac-

1-4244-0292-1/06/$20.00



(a) Three vector sequence



(b) Eight vector sequence with repetitions

Figure 1: Illustrating Stafan with biased inputs.

curacy obtained for a large industry design. Concluding remarks are included in Section 8

2 Motivation

For a s@1 fault at signal node G, Stafan [10] evaluates a detection probability per vector, $d0(G)$, which is then used to evaluate a cumulative detection probability, $D0(G)$, over N vectors as

$$D0(G) = 1 - k(1 - d0(G))^N \quad (1)$$

where k is a biasing factor. The above formula assumes that N vectors are statistically independent, as is obvious when k is assumed to be 1. The per vector detection probability, $d0(G) = c0(G) \times b0(G)$, is a product of two factors: (1) probability of fault excitation, $c0(G)$, and (2) conditional probability of fault effect propagation to an observation point, $b0(G)$, given that fault excitation has occurred. The latter is also referred to as zero-observability of gate G. For s@0 faults, the formula is analogous and involves quantities like $d1(G)$, $c1(G)$ and $b1(G)$:

$$D1(G) = 1 - k(1 - d1(G))^N \quad (2)$$

The probability of excitation of a s@1 fault at gate G is measured by the zero-controllability at G, estimated by counting the number of zeros and dividing by the total number of vectors N, i.e.,

$$c0(G) = zero\_count(G)/N \quad (3)$$

and, similarly,

$$c1(G) = one\_count(G)/N \quad (4)$$

The conditional fault effect propagation probabilities, $b0(G)$ and $b1(G)$, are evaluated backwards starting at primary outputs, and are initialized to 1 at every output that attains logic values 0 and 1, respectively. Otherwise, these probabilities are initialized to 0. For a three input OR gate, with inputs i_1, i_2 and i_3, and output o, the zero-observability at input i_1, $b0(i_1)$, is estimated as

$$b0(i_1) = b0(o) \times c0(o)/c0(i_1) \quad (5)$$

The factor $c0(o)/c0(i_1)$ is a measure of the conditional probability of the gate output having a logic value 0 given that input i_1 has a logic value 0. The above formula assumes that the conditional sensitization of input i_1 to output o is independent of the sensitization of the gate output to some primary output of the circuit. Similarly, the one-observability of input i_1 is approximated by

$$\begin{aligned} b1(i_1) &= b1(o) \times Pr(i_1, \overline{i_2}, \overline{i_3}|i_1) \\ &= b1(o) \times Pr(i_1, \overline{i_2}, \overline{i_3})/c1(i_1) \end{aligned}$$

By the law of total probability,

$$Pr(\overline{i_1}, \overline{i_2}, \overline{i_3}) + Pr(i_1, \overline{i_2}, \overline{i_3}) = Pr(\overline{i_2}, \overline{i_3})$$

However, $Pr(\overline{i_1}, \overline{i_2}, \overline{i_3}) = Pr(\overline{o}) = c0(o)$. Therefore,

$$Pr(i_1, \overline{i_2}, \overline{i_3}) = Pr(\overline{i_2}, \overline{i_3}) - c0(o) = S(i_1) - c0(o)$$

and

$$b1(i_1) = b1(o) \times (S(i_1) - c0(o))/c1(i_1) \quad (6)$$

where $S(i_1)$ is the sensitization probability of input i_1, measured by counting the number of cycles for which inputs i_2 and i_3 have non-controlling sensitizing values and dividing by the length of the input sequence. Similar formulas can be derived for other combinational gate types. Stafan requires such sensitization counts for each gate input, along with controllability counts for each gate output in the circuit.

Figure 1 shows a small circuit that illustrates the inaccuracies encountered in Stafan when input sequences are biased. Figure 1(a) shows all logic values for a sequence of three vectors. Also shown in the figure is the observability evaluation for logic 0 at $G6$. Since three vectors were applied, the detection probability for the

$G6$ s@1 fault evaluates to

$$D1(G6) = 1 - (1 - b0(G6) \times c0(G6))^3 = 0.53$$

Assuming the traditional threshold probability of 0.5 for detection, this fault is correctly estimated to be detected. Figure 1(b) shows the same circuit with 8 vectors, where the last five vectors are repetitions of the first vector. The observability at $G6$ now evaluates to 1/32, and the detection probability of the same fault is $1-(1-1/32)^8 = 0.22$. Therefore, the fault is now incorrectly determined not to be detected. If repetitions of the first vector are increased, the detection probability of this fault decreases even further.

To illustrate the errors incurred by Stafan, each of the ISCAS combinational benchmarks was simulated for two input sequences. For each circuit, a 100 clock cycle long "short sequence" was generated such that the logic values at all primary inputs were generated randomly. A "long sequence" was synthesized from the short sequence by a two-step process: (1) the primary input values for the first 50 clock cycles of the long sequence were identical to those of the short sequence and (2) each of the remaining 50 clock cycles of the short sequence was replicated a random number of times, varying from 1 to 100, and appended to the sequence obtained in step (1). These two sequences were simulated for all stuck-at faults in the circuits. Identical fault coverages were obtained from an exact fault simulator since all circuits used for this experiment were combinational. Coverage estimates were also obtained from Stafan for these two input sequences. Controllability and input sensitization measures for each gate were obtained during logic simulation, and observability measures were evaluated using formulas similar to Equations 5 and 6. The detection status for each fault was then estimated by using Equations 1 and 2. A fault was assumed to be detected if its detection probability exceeded 0.5. This experimental data is presented in Table 1.

The exact fault coverages for the two sequences are shown in column 2 of Table 1. The lengths of the two sequences are shown in columns 3 and 5, respectively. For the short sequences, coverage estimates are shown in column 4, while column 6 shows similar estimates for the long sequences. The difference between the estimates in columns 4 and 6 shows the sensitivity of coverage estimates to the vector length parameter N in Equations 1 through 4.

As evident from Table 1, fault coverage estimates vary significantly, even though test content is identical for the two sequences. Stafan's estimation algorithm assumes primary and internal gate inputs are random. Clearly, the second input sequence is a biased version of the first, and controllability measures like Equations 3 and 4 do not consider the vector-to-vector autocorrelation of an input signal. In addition, to considering such signal autocorrelation, a good estimation algorithm should consider signal correlation at inputs of internal gates. This paper presents a new estimation technique that accounts for signal correlation by evaluating an *effective vector length* (N_{eff}) for each sequence by observing the input combinations at each gate of the circuit.

Table 1: Identical coverage vector sets of varying lengths.

	exact	short sequence		long sequence	
circuit	cov %	len	est %	len	est %
(1)	(2)	(3)	(4)	(5)	(6)
c432	93.13	100	94.85	2200	98.28
c880	91.08	100	90.02	2280	91.30
c1355	87.93	100	18.74	2375	37.23
c1908	69.72	100	75.71	2760	81.73
c2670	77.28	100	63.49	2720	72.21
c3540	70.79	100	76.26	2650	84.39
c5315	91.72	100	76.59	2580	93.70
c6288	99.54	100	92.91	2410	98.76
c7552	84.42	100	74.54	2760	87.32

3 Analysis for Signal Monitoring

We briefly present some definitions before proceeding with the analysis. Most of the contents of this section are included here for completeness, otherwise the reader may find its details in a recent paper [2].

The *checkpoints* of a circuit consist of primary inputs and fanout signals [6]. A circuit is modeled as a directed graph where nodes of the graph represent gates, and edges correspond to signals in the circuit. Hence, ideas and concepts from graph theory are used extensively for structural analysis of circuits. In a directed graph, node A is a *dominator* of node B if every path from B to any output passes through A [3]. The set of nodes that dominates a given node is called a dominator set. Our definition of dominators differs from this classical graph theory definition. Even though every node is a dominator of itself in the classical sense, we exclude such consideration. In addition, we require that a dominator must have at least one input that is not reachable from the dominated gate. This eliminates single-input gate dominators and those that exist due to reconvergent fanouts only. If node A has a single fanout node B then B may dominate A only if B has other inputs not from A. A node is said to have a *trivial dominator set* if its immediate fanout is the only dominator. For a given node, a *non-trivial dominator set* necessarily includes at least one dominator that is not an immediate fanout point of the given node.

The *output sensitizing condition* for a gate and its dominator set denotes the condition that the gate output is at a specified value and all other off-path sensitizing values for its dominators have appropriate non-controlling values. The *input sensitizing condition* with respect to a specific input pin of a gate and its dominator set consists of the specified input having a given value

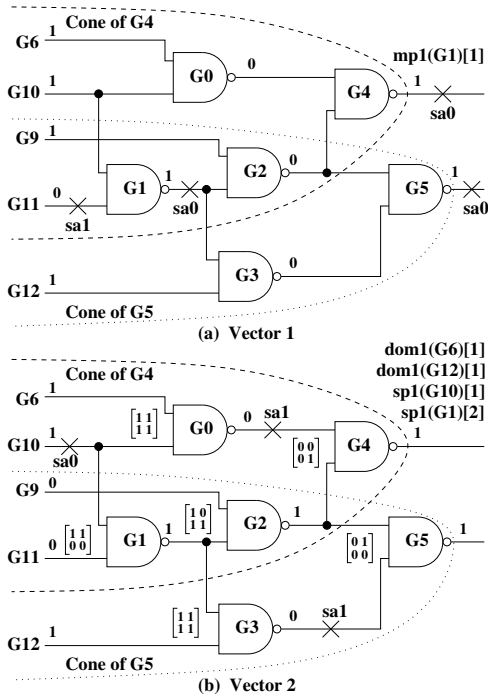


Figure 2: Simulation of c17 circuit for two vectors.

with all other inputs to the gate and off-path inputs to its dominators having non-controlling sensitizing values.

We consider a small circuit (c17) and a sequence of two vectors to illustrate how the algorithm determines signal conditions to monitor. There are three distinct steps to finding fault detection probability: (1) a preprocessing step consisting of structural analysis, (2) monitoring of signal conditions during simulation and (3) post-processing of results and evaluation of fault detection probability. The first two stages are outlined in this section. The final step for this example is presented in Section 5 after the algorithm is given in Section 4.

3.1 Structural Analysis

Structural graph analysis for finding dominators [11] and reconvergence points [12, 13, 15] has been used in test generation systems. We use a similar analysis to monitor signal conditions at the end of each cycle in a vector sequence. These conditions are derived for single output circuits. For multi-output circuits, the conditions are obtained separately from each cone of logic that drives a primary output. These signal conditions are *necessary*, but not *sufficient*, for fault detection.

For each gate, all input value combinations are stored in a table. These input combinations are analyzed once simulation of the entire sequence is complete. Along with gate input combinations, each logic cone is analyzed separately and dominators for each *non-fanout* gate to the cone output are obtained. Such dominators found for non-fanout gates are either trivial or non-trivial. For the circuit of Figure 2, gate $G0$ has a trivial dominator set $\{G4\}$, while $G6$ has a non-trivial dominator set $\{G0, G4\}$. The other dominated gates are $G3$ and $G12$, with dominator sets $\{G5\}$ (trivial) and $\{G3, G5\}$ (non-trivial), respectively. Since this dominator analysis is performed for each cone, additional dominators are found for gates $G1$ and $G2$. For the cone of $G4$, $G2$ has a trivial dominator set $\{G4\}$, while $G1$ has a non-trivial dominator set $\{G2, G4\}$. In the cone of $G5$, $G2$ has a trivial dominator set $\{G5\}$.

Table 2: Sensitizing conditions for dominators in c17.

$dom0(G6)[1]$	=	{G6=0,G10=1, G2=1}
$dom0(G12)[1]$	=	{G12=0, G1=1, G2=1}
$dom0(G9)[1]$	=	{G9=0,G1=1, G3=1}
$dom0(G9)[2]$	=	{G9=0, G1=1, G0=1}
$dom0(G1)[1]$	=	{G1=0, G9=1, G0=1}
$dom0(G10)[1]$	=	{G10=0, G11=1, G9=1, G3=1}
$dom1(G6)[1]$	=	{G6=1,G10=1, G2=1}
$dom1(G12)[1]$	=	{G12=1, G1=1, G2=1}
$dom1(G9)[1]$	=	{G9=1,G1=1, G3=1}
$dom1(G9)[2]$	=	{G9=1, G1=1, G0=1}
$dom1(G1)[1]$	=	{G1=1, G9=1, G0=1}
$dom1(G10)[1]$	=	{G10=1, G11=1, G9=1, G3=1}

For gates with trivial dominator sets, input sensitizing conditions are monitored explicitly, unless an input is a checkpoint. For non-trivial dominators, only output sensitizing conditions are monitored. Output sensitizing conditions from trivial dominators are ignored because they are obtained automatically by monitoring input states for the dominating gate. For the circuit of Figure 2, all inputs of both gates $G0$ and $G3$ are checkpoints and are skipped. For $G6$, the output sensitizing condition is the simultaneous occurrence of {G6=v, G10=1, G2=1} for $v \in \{0, 1\}$. For $G12$, the non-trivial condition consists of {G12=v, G1=1, G2=1} for $v \in \{0, 1\}$. In the cone of $G4$, the trivial dominator set for $G2$ yields two conditions for its second input: {G1=v, G9=1, G0=1} for $v \in \{0, 1\}$. These conditions are considered because $G1$ is not a checkpoint in this cone. The non-trivial dominator condition for $G9$ yields {G9=v, G1=1, G0=1}. For the cone of $G5$, gates $G2$ and $G9$ have dominator sets $\{G5\}$ and $\{G2, G5\}$. Both inputs for $G2$ are checkpoints for the cone and are ignored. The output sensitizing conditions for $G9$ and its dominator set are {G9=v, G1=1, G3=1}, for both $v \in \{0, 1\}$. Table 2 lists the conditions that are obtained for both signal values at each gate. Multiple conditions at a gate are shown with an index in square brackets ([ ]).

A dominator set {G1, G2, G4} exists for gate $G11$ in the cone of $G4$. However, no dominator exists for this gate in the cone of $G5$. If no conditions are obtained for a gate from a cone, all conditions for that gate obtained from other cones are also dropped from further consideration. A fault that appears in multiple cones is considered to have zero detection probability if the relevant condi-

Step 1: Partition circuit into cones, one for each output. In each cone, for each gate G, initialize dom0(G), dom1(G), sp0(G), sp1(G), mp0(G) and mp1(G) to 1.
Step 2: For each non-fanout gate in a cone, classify dominators as trivial and non-trivial:
(a) For trivial dominators, list input sensitizing conditions for both logic values if input is not a checkpoint of the cone
(b) For non-trivial dominators, list output sensitizing conditions for both logic values.
Step 3: For fanout gates that have reconvergent paths with same parity, list multiple path activation conditions for both logic values.
Step 4: For fanout gates that have reconvergent paths with different parity, list single path activation conditions for both logic values.
Step 5: If Steps (2)-(4) yield no condition for G in a given cone, drop all conditions for G from all other cones. Initialization in Step (1) guarantees faults local to G are not considered undetectable.

Figure 3: Algorithm for selection of conditions.

Table 3: Different parity sensitizing conditions for $G10$.

$sp0(G10)[1] = \{G10 = 0, G6 = 1, G2 = 1, G9 = 0 \| G11 = 0\}$
$sp0(G10)[2] = \{G10 = 0, G11 = 1, G9 = 1, G0 = 1, G6 = 0\}$
$sp1(G10)[1] = \{G10 = 1, G6 = 1, G2 = 1, G9 = 0 \| G11 = 0\}$
$sp1(G10)[2] = \{G10 = 1, G11 = 1, G9 = 1, G0 = 1, G6 = 0\}$

tions in *all* cones remain unsatisfied during simulation. Gates $G10$ and $G1$ also have conditions from one cone only. However, these are not dropped because they are fanout points in other cones, and additional conditions due to reconvergence in the other cone are generated, as explained below.

Fanout gates are analyzed specific to each output cone that contains them. These fanout gates in a cone are origination points for reconvergent paths. Some fanout gates in the circuit may have no reconvergent fanout in any cone and are not considered in this step, e.g., $G2$. Different reconvergent paths may have different parity and fault propagation requires that paths with different parity be not simultaneously sensitized. However, along paths that have the same parity, simultaneous fault effect propagation may occur.

For the cone of $G4$, $G10$ has reconvergent paths with different parity. We denote the two sensitizing conditions for paths originating at $G10$ by $sp0(G10)[1]$ ($sp1(G10)[1]$) and $sp0(G10)[2]$ ($sp1(G10)[2]$) for logic value 0 (1). These are shown in Table 3. The notation "sp" is used for single paths, while "mp" denotes multiple paths, as explained in the following paragraph.

For $sp0(G10)[1]$, $\{G9 = 0 \| G11 = 0\}$ represents the disabling condition for propagation paths with same parity. For the cone of $G5$, there are reconvergent paths with the same parity from $G1$. Considering that it is possible to activate any subset of these paths, including all of them simultaneously, Table 4 shows the conditions for $G1$.

For a given gate G, if none of the conditions $dom0(G)$, $sp0(G)$ or $mp0(G)$ ($dom1(G)$, $sp1(G)$ or $mp1(G)$) is ever satisfied during simulation, the $sa1$ ($sa0$) fault at the output of G will remain undetected. In addition, all faults whose propagation requires gate G to have a value 0 (1) also remain undetected. The algorithm for selection of conditions to be monitored is summarized in Figure 3.

Table 4: Same parity sensitizing conditions for $G1$.

$sp0(G1)[1]$	=	$\{G1 = 0, G9 = 1, G3 = 1\}$
$sp0(G1)[2]$	=	$\{G1 = 0, G12 = 1, G2 = 1\}$
$mp0(G1)[1]$	=	$\{G1 = 0, G9 = 1, G12 = 1\}$
$sp1(G1)[1]$	=	$\{G1 = 1, G9 = 1, G3 = 1\}$
$sp1(G1)[2]$	=	$\{G1 = 1, G12 = 1, G2 = 1\}$
$mp1(G1)[1]$	=	$\{G1 = 1, G9 = 1, G12 = 1\}$

3.2 Signal Monitoring in Example

Figure 2 shows a two vector simulation sequence of the example circuit, along with specific vectors where these conditions are *first* satisfied. Also shown in the figure are the gate input combinations that are observed after simulation of each vector. Once simulation of a subsequence is complete, the set of input value combinations for each gate and the conditions of Section 3.1 that are satisfied are analyzed. Table 5 shows the values of conditions after simulation of the second vector is complete. For fanout points, the $sp0$ and $sp1$ entries in the table show a list of entries, one for each fanout branch. The algorithm for fault detection prediction using these satisfied logic conditions is presented in Section 4, following which we revisit this example in Section 5.

4 Post-Processing

There are three main differences between our algorithm and Stafan: (1) The sequence length, N, in Equations 1 and 2 is modified and a new value, N_{eff}, is used instead. (2) The measurement of controllability measures, $c0$ and $c1$ in Equations 3 and 4, is modified and (3) Results of signal monitoring are used for observability evaluation, which is performed separately for each cone in the circuit. These differences are outlined in this section.

Table 5: Signal monitoring result after second vector.

signal	cone of $G4$						cone of $G5$					
	$dom0$	$sp0$	$mp0$	$dom1$	$sp1$	$mp1$	$dom0$	$sp0$	$mp0$	$dom1$	$sp1$	$mp1$
G4	1	1	1	1	1	1	-	-	-	-	-	-
G5	-	-	-	-	-	-	1	1	1	1	1	1
G0	1	1	1	1	1	1	-	-	-	-	-	-
G2	1	1	1	1	1	1	1	1	1	1	1	1
G3	-	-	-	-	-	-	1	1	1	1	1	1
G1	0	1	1	0	1	1	1	0,0	0	1	0,1	1
G9	0	1	1	0	1	1	0	1	1	0	1	1
G11	1	1	1	1	1	1	1	1	1	1	1	1
G10	1	0,0	1	1	1,0	1	0	1	1	0	1	1
G6	0	1	1	1	1	1	-	-	-	-	-	-
G12	-	-	-	-	-	-	0	1	1	1	1	1

4.1 Effective Vector Lengths (N_{eff})

As demonstrated by the results of Table 1 in Section 2, the fault detection probabilities evaluated by Equations 1 and 2 may vary substantially for non-random input sequences. The input combinations at every internal gate are monitored during logic simulation. This can be implemented efficiently by monitoring events that are processed by the signal scheduler in any logic simulator, and updating a table of previously seen input combinations at each fanout of the scheduled gate. Since fanout processing is an inherent function of the signal scheduler, this additional code does not increase the runtime complexity of the scheduler. The effective length of a sequence, N_{eff} is increased whenever a new input combination is observed at any gate in the circuit.

Once N_{eff} is obtained, Equations 1 and 2 are modified to

$$D0(G) = 1 - k(1 - d0(G))^{N_{eff}} \quad (7)$$

$$D1(G) = 1 - k(1 - d1(G))^{N_{eff}} \quad (8)$$

where $d0(G)$ and $d1(G)$ are obtained from controllabilities and observabilities that are evaluated as explained in Sections 4.2 and 4.3.

4.2 Controllability Evaluation

Controllability measures are implemented by counters, one for each gate in the circuit. For a given gate, this counter counts the number of ones seen during all clock cycles that results in an increment of N_{eff}. As shorthand notation, we denote this counter as $\kappa(G)$ for gate G. Similarly, $\kappa(\overline{G}) = N_{eff} - \kappa(G)$. For clock cycles that do not result in any new input state at any gate, all controllability counters are kept unchanged. Equations 3 and 4 are replaced by

$$c0(G) = zero\_count(G)/N_{eff} = \kappa(\overline{G})/N_{eff} \quad (9)$$

$$c1(G) = one\_count(G)/N_{eff} = 1 - c0(G) \quad (10)$$

4.3 Observability Evaluation

For each gate in the circuit, the set of input combinations that is observed is stored in a table, along with a count for each combination. This table and the counts for each input state are updated whenever the parameter N_{eff} is incremented. For an OR gate G with output o, and inputs i_1, i_2 and i_3, the notation $\kappa(i_1, i_2, i_3)$ denotes the counter for the input combination $i_1 = 1, i_2 = 1, i_3 = 1$. Also, note that $\kappa(\overline{i_1}, \overline{i_2}, \overline{i_3}) = \kappa(\overline{o})$. In terms of these counters, the sensitization probability used in Equation 6 can be simplified as

$$S(i_1) = Pr(\overline{i_2}, \overline{i_3}) = (\kappa(\overline{i_1}, \overline{i_2}, \overline{i_3}) + \kappa(i_1, \overline{i_2}, \overline{i_3}))/N_{eff}$$

Similarly, for a three input AND gate,

$$S(i_1) = Pr(i_2, i_3) = (\kappa(\overline{i_1}, i_2, i_3) + \kappa(i_1, i_2, i_3))/N_{eff}$$

The observability measures, $b0(G)$ and $b1(G)$, are evaluated backwards starting at primary outputs. However, unlike Stafan, these probabilities are evaluated separately for each cone in the circuit, as explained in the following subsections.

4.3.1 Propagation at Non-Fanout Inputs

If input i_1 of OR gate, with output o, is not a fanout stem, $b0(i_1)$ is evaluated as

$$b0(i_1) = b0(o) \times (\kappa(\overline{o})/\kappa(\overline{i_1})) \times dom0(i_1) \quad (11)$$

This equation is similar to Equation 5, except for the additional multiplication factor $dom0(i_1)$. This factor resets $b0(i_1)$ if dominator condition for this input remains unsatisfied. The observability for logic value 1 is evaluated as follows:

$$b1(i_1) = b1(o) \times ((S(i_1) - \kappa(\overline{o}))/\kappa(i_1)) \times dom1(i_1) \quad (12)$$

Similar formulas can be derived for an AND gate.

4.3.2 Propagation at Stem Inputs

Fanout stems can vary depending on the inversion parity of reconvergent paths. If reconvergent paths have different parity, fault propagation cannot occur simultaneously because such fault effects cancel out at the point of reconvergence. For paths that have the same parity, simultaneous fault effect propagation along multiple paths is possible and needs to be modeled.

Before proceeding further, we introduce some notation. Assume $E_i, 1 \leq i \leq M$ are independent events, and E is the composite event defined by set union, i.e.,

$$E = \bigcup_{i=1}^{M} E_i$$

We denote the probability of event E by the notation

$$Pr(E) = Ind\{Pr(E_i),\ 1 \leq i \leq M\}$$

where $Ind\{\}$ is the joint probability of independent events. For $M = 2$,

$$\begin{aligned} Ind\{Pr(E_1), Pr(E_2)\} \quad &= \quad Pr(E_1) + Pr(E_2) \\ &\quad -Pr(E_1) \times Pr(E_2) \end{aligned}$$

Assume stem s has fanout branches $f_i, 1 \leq i \leq M$, all reconverging to gate r with the same parity. At stem s,

$$b0(s) = Ind\{b0(r) \times mp0(s), b0(f_i), 1 \leq i \leq M\} \quad (13)$$
$$b1(s) = Ind\{b1(r) \times mp1(s), b1(f_i), 1 \leq i \leq M\} \quad (14)$$

If all fanout branches reconverge at r with a different parity than that of s, the above formulas are modified to

$$b0(s) = Ind\{b1(r) \times mp0(s), b0(f_i), 1 \leq i \leq M\} \quad (15)$$
$$b1(s) = Ind\{b0(r) \times mp1(s), b1(f_i), 1 \leq i \leq M\} \quad (16)$$

If reconvergence occurs at gate r with different parity, the single path propagation predicates for each fanout are used instead.

$$b0(s) = Ind\{(b0(f_i) \times sp0(f_i)), 1 \leq i \leq M\} \quad (17)$$
$$b1(s) = Ind\{(b1(f_i) \times sp1(f_i)), 1 \leq i \leq M\} \quad (18)$$

4.4 Statistical Coverage and Bounds

Suppose that among the N_{eff} vectors that can possibly contribute to detection of a fault, exactly one vector actually detects certain fault under consideration. Then the per-vector detection probability ($d0$ or $d1$) for that fault will be $1/N_{eff}$. Using Equation 7 or 8, this will give the cumulative detection probability ($D0$ or $D1$) for N_{eff} vectors as,

$$\lim_{N_{eff} \to \infty} 1 - (1 - \frac{1}{N_{eff}})^{N_{eff}} = 1 - \frac{1}{e} = 0.63 \quad (19)$$

Table 6: Observabilities after second vector.

signal	$\kappa(G)$	Cone of $G4$		Cone of $G5$	
		$b0$	$b1$	$b0$	$b1$
G4	2	0.0	1.0	–	–
G5	2	–	–	0.0	1.0
G0	0	0.5	0.0	–	–
G2	1	0.0	0.0	0.0	0.0
G3	0	–	–	0.5	0.0
G1-G2	2	–	–	0.0	0.0
G1-G3	2	–	–	0.0	0.5
G1	2	0.0	0.0	0.0	1.0
G6	2	0.0	0.5	–	–
G10-G0	2	0.0	0.5	–	–
G10-G1	2	0.0	0.0	–	–
G10	2	0.0	0.5	0.0	0.0
G9	1	0.0	0.0	0.0	0.0
G11	0	0.0	0.0	1.0	0.0
G12	2	–	–	0.0	0.5

This is a valid approximation for large values of N_{eff}. Here as well as in the next section, we have assumed $k = 1.0$ for the biasing factor in Equations 1, 2, 7 and 8. Thus, we will use 0.63 as a *threshold* to decide whether or not a fault should be counted as detected. When the cumulative detection probability of a fault equals or exceeds 0.63, we will assume that the fault has been detected by one or more vectors and we count it as detected while computing the statistical coverage.

We determine statistical bounds on coverage by using a threshold close to 0.0 (any fault with non-zero probability can be potentially detected) for an upper bound, and a threshold close to 1.0 for a lower bound. We have used a threshold of 0.01 for statistical upper bound and found it to be almost the same as the exact upper bound [2]. For lower bound estimate, we used a threshold of 0.99.

5 Post-Processing of Example

For the example of Figure 2, the gate input states seen after the second vector are shown in Figure 2(b). The gate dominator and path activation conditions that are satisfied are shown in Table 5. Since new input combinations are observed in the second vector, the value of the N_{eff} parameter is 2. Table 6 shows the observability measures that are obtained for each gate in the two logic cones of the circuit.

For the 22 equivalence collapsed faults in the circuit, Table 7 lists the controllabilities and observabilities of fault sites in columns 4 and 5, respectively. For a signal that is common to both logic cones, the higher value of observability is shown in Column 5. Using $N_{eff} = 2$ and biasing factor $k = 1$ in Equations 7 and 8, the detection probabilities obtained are shown in Column 6. Using a detection probability threshold of 63% the first seven faults are estimated to be detected by this two-vector

Table 7: Detectability of individual faults.

id (1)	location (2)	type (3)	cont (4)	obs (5)	det (6)
1	$G4$	$s@0$	1.0	1.0	1.0
2	$G5$	$s@0$	1.0	1.0	1.0
3	$G1$	$s@0$	1.0	1.0	1.0
4	$G11$	$s@1$	1.0	1.0	1.0
5	$G0$	$s@1$	1.0	0.5	0.75
6	$G3$	$s@1$	1.0	0.5	0.75
7	$G10$	$s@0$	1.0	0.5	0.75
8	$G4$	$s@1$	0	0	0
9	$G6$	$s@1$	0	0	0
10	$G5$	$s@1$	0	0	0
11	$G2$	$s@0$	0.5	0	0
12	$G9$	$s@1$	0.5	0	0
13	$G10 - G0$	$s@1$	0	0	0
14	$G10 - G1$	$s@1$	0	0	0
15	$G10$	$s@1$	0	0	0
16	$G2 - G4$	$s@1$	0.5	0	0
17	$G2 - G5$	$s@1$	0.5	0	0
18	$G2$	$s@1$	0.5	0	0
19	$G1 - G2$	$s@1$	0	0	0
20	$G1 - G3$	$s@1$	0	0	0
21	$G1$	$s@1$	0	0	0
22	$G12$	$s@1$	0	0	0

sequence. This matches with exact fault simulation.

6 Upper-Bounding Comparison

We illustrate the difference between coverage estimation and upper-bounding [2] using the circuit of Figure 4. This circuit is the logic cone of gate $G5$ in circuit $c17$ of Figure 2. Logic values for a three vector sequence are shown in that figure. As explained in Section 3.1, this circuit has two reconvergent paths with the same parity and originating at $G1$. The multiple path activation conditions for both logic values 0 and 1 remain unsatisfied in these three vectors. Since new input states are observed during each of these vectors, N_{eff} is set to 3 for the sequence. Furthermore, the value of N_{eff} will stay unchanged should any of these vectors were to be repeated. The figure also shows the zero-observabilities at gates $G11$ and $G3$, and one-observabilities at gates $G1$ and $G5$. These observabilities are required to evaluate the detection probability of the s@1 fault at $G11$. Since zero-controllability at this gate is 1, the detection probability of the fault is $1 - (1 - 1/9)^3 = 0.22$. Using a detection threshold of 0.63, this fault is estimated to be undetected, which matches with exact fault simulation.

For a given gate G, upper-bounding [2] evaluates propagation predicates, $oneProp(G)$ and $zeroProp(G)$, that denote the possibility of logic values 1 and 0, respectively, to propagate from G to a primary output [2]. For an output that attains logic value 1 (0) sometime during simulation, $oneProp(G)$ ($zeroProp(G)$) is assigned a

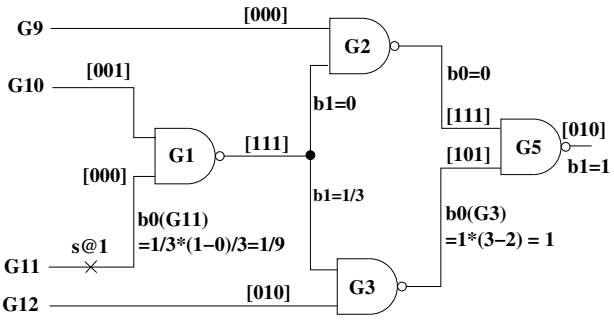


Figure 4: Upper-bounding comparison.

value 1. Otherwise, this value is set to 0. Using backward traversal from the outputs, these predicates are evaluated at gate inputs. Unlike observability values that are probabilities, these predicates denote the possibility of logic value propagation through the circuit. For this circuit, $zeroProp$ evaluates to 1 (0) at gate $G3$ ($G2$). Using stem analysis, $oneProp$ evaluates to 1 at stem $G1$. At gate $G11$, $zeroProp$ evaluates to 1 because the input combination $(G10 = 1, G11 = 0)$ is observed at gate $G1$. Hence, upper-bounding would consider this fault as a potentially detected fault.

As can be seen by the preceding example, the algorithm in this paper eliminates some false positive errors that arise during the upper-bounding process. However, some false negative errors also result due to faults that have low probability of detection. When compared to the upper-bounding estimates [2], fewer false positive errors and slightly higher false negative errors result in aggregate coverage estimates that are now closer to exact fault simulation. This effect is more pronounced in circuits with high combinational depth.

7 Results

The experimental setup chosen for verification of the algorithm is identical to that described in Section 2. Each of the ISCAS combinational benchmarks was simulated for a randomly input "short sequence" 100 cycles long. A "long sequence" was obtained by concatenating the first 50 vectors, and repeating each of the remaining 50 vectors randomly 1 to 100 times. The two sequences thus have identical fault coverages. The results of these simulation are presented in Table 8. Column 2 shows the exact coverage obtained from a fault simulator. For the short sequence, columns 3 and 4 show sequence length and coverage estimates obtained from Stafan. Similar data for the long sequence is presented in columns 5 and 6. The numbers in columns 3 through 6 are identical to those of Table 1. Columns 7 through 10 show results obtained by the algorithm of this paper. Column 7 shows the effective length obtained for these sequences. Column 8 shows the coverage estimates obtained, while the false positive and false negative errors are shown in

Table 8: Improved coverage estimates for combinational benchmarks.

circuit (1)	exact cov. (%) both sequences (2)	Stafan (short) len (3)	Stafan (short) est (%) (4)	Stafan (long) len (5)	Stafan (long) est (%) (6)	new algorithm (both sequences) N_{eff} (7)	est (%) (8)	pos (9)	neg (10)	CPU time over logic sim. (11)
c432	93.15	100	94.85	2200	98.28	89	92.94	1.34	1.15	33%
c880	91.08	100	90.02	2280	91.30	63	90.13	1.70	2.65	20%
c1355	87.93	100	18.74	2375	37.23	50	88.37	0.89	0.44	22%
c1908	69.79	100	75.71	2760	81.73	82	72.94	3.36	0.21	31%
c2670	77.28	100	63.49	2720	72.21	74	77.28	1.57	1.57	37%
c3540	70.79	100	76.26	2650	84.39	100	69.53	1.20	2.49	31%
c5315	91.74	100	76.59	2580	93.70	100	92.24	2.92	2.41	35%
c6288	99.74	100	92.91	2410	98.76	46	99.74	0.00	0.00	16%
c7552	84.47	100	74.54	2760	87.32	89	86.19	2.73	1.00	52%

columns 9 and 10, respectively. The CPU time increase over "plain" logic simulation is shown in column 11. This is mainly due to the signal monitoring [2]. In addition, preprocessing and postprocessing, both of which are independent of the vector length are negligible.

An additional experiment was performed to verify that estimates obtained by this algorithm closely track exact fault coverages at all intermediate vectors in a sequence. Both Stafan and upper bound estimates were also obtained and compared to the new statistical estimate. Additionally, two other statistical estimates were obtained by taking the detection probability thresholds as 1% and 99% instead of the default value of 63%. A long sequence was constructed by repeatedly concatenating identical copies of the short random sequence of 100 cycles described in the previous paragraph. The results obtained from this experiment are shown for c2670 in Figure 5. Since the new sequence obtained is long, the x-axis of the graph is plotted using a logarithmic scale (base 10). After 100 vectors, the coverage from exact simulation remains constant at 77.28%. The upper bound coverage was found to be 79.94%. All statistical estimates, using 1%, 63% and 99% detection probability thresholds, and the upper bound estimate also stay constant beyond 100 vectors at their respective values attained at the 100th vector. The Stafan estimate reaches 63.49% at vector 100, but continues to rise to 72.3% at vector 8000. Statistical estimates using 1%, 63% and 99% thresholds attained final values of 79.94%, 77.28% and 62.14%, respectively. The statistical estimates by the algorithms presented in this paper correct Stafan's error that causes the coverage to increase as the short sequence of 100 vectors is repeated.

Evident in the above graph is the difference between the statistical (63% threshold) estimate and the upper bound estimates (1% threshold or [2]). This difference as a function of vector length is labeled as "upper bound – new estimate" in Figure 5. Even though this difference after 100 vectors was only 1.38% ($= 79.94 - 78.56\%$), its highest value 12.43% was attained after 8 vectors when the exact coverage was 53.1%. At this point, the statistical estimate was 53.54%, while the upper bound estimate was 65.97%. The Stafan coverage underestimates the true coverage at all intermediate points of the sequence, even though it continues to increase gradually after 100 vectors.

An industry example: As mentioned in Section 1, fault simulation can be very resource intensive. Figure 6 shows simulation data for a design with 180K gates that was fault simulated with 42 functional sequences. The vertical lines on the x-axis show fault coverages for individual sequences, while the cumulative exact coverage is shown as a curve. Also included in the figure are cumulative upper bound [2] and the statistical estimate. The time required to fault simulate the longest sequence was about 7 days. Assuming the availability of multiple machines, it takes a week to compute the cumulative coverage of all input sequences. Both upper bound and statistical estimates were obtained within 3 hours for each sequence, about the time required for logic simulation. The exact cumulative coverage of the 42 functional sequences was 58.76%, while upper bound and statistical estimates were 69.38% and 63.09%, respectively.

To attain acceptable fault coverage, many more functional sequences are required. Our experiment was designed to complete in acceptable time. Many designers like to experiment with different sets of scan elements either to maximize fault coverage or remove scan elements from critical timing paths. Experiments to change the list of observable points are virtually impossible if one has to rely solely on fault simulation.

8 Conclusion

This paper clearly illustrates the improvement in stuck fault coverage estimation when the proposed algorithm is used for non-random and functional input vectors. In combinational circuits, inputs to a gate may be correlated due to internal fanout reconvergence. In sequential circuits, even for random inputs, latch outputs that drive combinational logic are often correlated. The ideas presented in this paper improve fault coverage estimation in all of these cases. However, sequential circuit coverage estimation still remains an area for

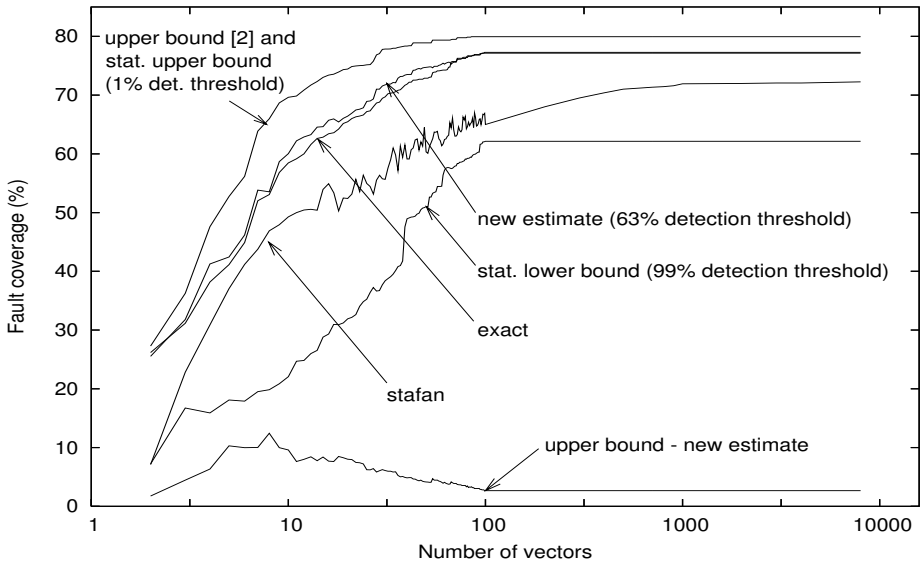


Figure 5: Exact, upper bound [2], and statistical coverages of c2670 by a long deterministic sequence.

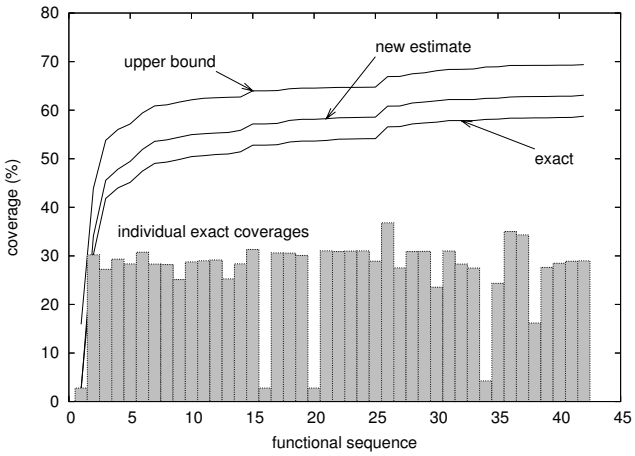


Figure 6: Example estimation for large design.

active research. Some algorithms that further improve coverage estimation in sequential designs will be presented in the future.

References

[1] M. Abramovici, P. R. Menon, and D. T. Miller, "Critical Path Tracing: An Alternative to Fault Simulation," *IEEE Design & Test of Computers*, vol. 1, no. 1, pp. 83–93, Feb. 1984.

[2] V. D. Agrawal, S. Bose, and V. Gangaram, "Upper Bounding Fault Coverage by Structural Analysis and Signal Monitoring," in *Proc. 24th IEEE VLSI Test Symp.*, 2006, pp. 88–93.

[3] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Design and Analysis of Computer Algorithms*. Reading, Massachusetts: Addison-Wesley, 1974.

[4] K. J. Antriech and M. Schulz, "Accelerated Fault Simulation and Fault Grading in Combinational Circuits," *IEEE Trans. CAD*, vol. 6, no. 9, pp. 704–712, Sept. 1987.

[5] D. B. Armstrong, "A Deductive Method of Simulating Faults in Logic Circuits," *IEEE Trans. Computers*, vol. C-21, pp. 464–471, May 1972.

[6] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Boston: Springer, 2000.

[7] W. T. Cheng and M. L. Yu, "Differential Fault Simulation for Sequential Circuits ," *Jour. Electronic Testing: Theory and Applications*, vol. 1, no. 1, pp. 7–13, Feb. 1990.

[8] S. Gai and P. L. Montessoro, "CREATOR: New Advanced Concept in Concurrent Simulation," *IEEE Trans. CAD*, vol. 13, no. 6, pp. 786–795, June 1994.

[9] S. Gai, P. L. Montessoro, and F. Somenzi, "MOZART: A Concurrent Multilevel Simulator ," *IEEE Trans. CAD*, vol. 7, no. 9, pp. 1005–1016, Sept. 1988.

[10] S. K. Jain and V. D. Agrawal, "Statistical Fault Analysis," *IEEE Design & Test of Computers*, vol. 2, no. 1, pp. 38–40, Feb. 1985.

[11] T. Kirkland and M. R. Mercer, "A Topological Search Algorithm for ATPG," in *Proc. 24th Design Automation Conf.*, June 1987, pp. 502–508.

[12] W. Kunz and D. K. Pradhan, "Accelerated Dynamic Learning for Test Pattern Generation in Combinational Circuits," *IEEE Trans. CAD*, vol. 12, pp. 684–694, May 1993.

[13] W. Kunz and D. K. Pradhan, "Recursive Learning: A New Implication Technique for Efficient Solutions to CAD Problems - Test, Verification and Optimization," *IEEE Trans. CAD*, vol. 13, pp. 1143–1158, Sept. 1994.

[14] T. M. Niermann, W. T. Cheng, and J. H. Patel, "PROOFS: A Fast Memory Efficient Sequential Circuit Fault Simulator," *IEEE Trans. CAD*, vol. 11, no. 2, pp. 198–207, Feb. 1992.

[15] M. H. Schulz, E. Trischler, and T. M. Sarfert, "SOCRATES: A Highly Efficient Automatic Test Pattern Generation System," *IEEE Trans. CAD*, vol. 7, pp. 126–37, Jan. 1988.

[16] E. G. Ulrich and T. Baker, "Concurrent Simulation of Nearly Identical Digital Networks," *Computer*, vol. 7, pp. 39–44, Apr. 1974.