Circuit Lines for Guiding the Generation of Random Test Sequences for Synchronous Sequential Circuits

Irith Pomeranz[1]
School of Electrical & Computer Eng.
Purdue University
W. Lafayette, IN 47907, U.S.A

and

Sudhakar M. Reddy[2]
Electrical & Computer Eng. Dept.
University of Iowa
Iowa City, IA 52242, U.S.A.

Abstract - A procedure proposed earlier for improving the fault coverage of a random primary input sequence modifies the input sequence so as to avoid repeated synchronization of state variables. We show that in addition to the values of state variables, it is also important to consider repeated setting of other lines to the same values. A procedure and experimental results are presented to demonstrate the improvements in fault coverage of random primary input sequences when the values of selected lines are considered.

I. Introduction

Test sequences for synchronous sequential circuits [1]-[5] are used for the purpose of manufacturing testing, where they may detect defects that are not detected by scan-based tests [6], and for simulation-based design verification. For both applications, the ability to achieve high fault coverage using random primary input sequences can facilitate the test generation process significantly. However, it is well-known that random primary input sequences achieve low stuck-at fault coverage for synchronous sequential circuits. The low fault coverage of random primary input sequences prevents their use as alternatives to deterministic test sequences. Such alternatives are important due to the high complexity of deterministic test generation for synchronous sequential circuits [1]-[5]. In [7] it was shown that the following effect plays an important role in causing the low fault coverage of random input sequences.

Subsets of state variables may be repeatedly synchronized, i.e., forced to the same values, by a random primary input sequence. This is related to the existence of input cubes that synchronize subsets of state variables. An input cube c is said to *synchronize* a subset of state variables $S(c)$ if applying c to the primary inputs, when the circuit is in the all-unspecified state, results in the specification of the state variables in $S(c)$ one time unit later. For example, we consider a circuit with four primary inputs and six state variables. Suppose that an input cube c = 0xxx synchronizes four of the state variables such that the next state under c is 0000xx when the present state is xxxxxx (x stands for an unspecified value). In a random primary input sequence, approximately half of the input vectors may be covered by c = 0xxx, or have a 0 on the first input. Therefore, the circuit will enter state 0000xx (or a state that it covers) at approximately half of the time units. With four of the six state variables at 0 in half of the time units, a low fault coverage is likely.

This issue was addressed in [7] by a two-phase process. In the first phase, input cubes that synchronize some or all of the state variables of the circuit are identified. The set of input cubes that synchronize subsets of state variables is denoted by C. In the second phase of the process, a random primary input sequence T is generated (one or more sequences can be considered in the same way). The sequence T is modified so as to eliminate the appearances in T of input cubes from C. Specifically, if an input vector t of T is covered by cubes from C, t is modified by complementing values that match the values under the cubes from C. Experimental results presented in [7] show that the fault coverage achieved by T is increased significantly once the appearances of cubes from C are eliminated. We review this process in more detail in Section II.

The focus in [7] was on state variables that may be synchronized repeatedly by a random primary input sequence. In this work we note that setting other lines of the circuit to the same values repeatedly during a random primary input sequence may also limit the fault coverage achievable by the input sequence. For example, if a line g assumes the value 0 repeatedly under a random primary input sequence T, T may not detect faults that require g = 1 for activation or propagation. To address this issue we identify a set of lines G that may be set repeatedly to the same values by a random primary input sequence. We include the next-state variables in G in order to ensure that repeated synchronization is considered as well. We modify the computation of the set of input cubes C (the first phase of the procedure from [7]) to consider all the lines in G. We then use the set of input cubes C to modify a random primary input sequence T (the second phase of the procedure from [7]). We thus prevent T from repeating the same values on the lines in G, which

1. Research supported in part by SRC Grant No. 2004-TJ-1244.
2. Research supported in part by SRC Grant No. 2004-TJ-1243.

978-1-4244-1922-7/08/$25.00 ©2008 IEEE

includes the state variables as well as other lines for which repeated setting to the same value may be an issue.

In Section III we define the set of lines G whose values should be considered during the computation of the set of input cubes C. We then describe the changes made to the procedure from [7] for computing C. Experimental results are presented in Section IV.

II. Review of the Procedure from [7]

In this section we review the procedure from [7].

A. Computing the Set of Input Cubes C

In [7], the set of input cubes C is constructed by considering a preselected number M of random primary input vectors. Every input vector v_i is applied to the primary inputs of the fault free circuit when the circuit is in the all-unspecified state. Let $S(v_i)$ be the set of state variables that become specified under v_i one time unit later. If $S(v_i) \neq \phi$, an attempt is made to unspecify the input values under v_i one at a time. Let v_i^j be the vector obtained from v_i by unspecifying the value of input j. Let $S(v_i^j)$ be the set of state variables that become specified under v_i^j. If $S(v_i^j) = S(v_i)$, v_i is replaced with v_i^j, thus accepting to unspecify input j; otherwise, input j remains specified under v_i. The inputs are considered in a random order to ensure that different input cubes may be obtained even if the same vector v_i is used as a starting point. After all the inputs have been considered, the incompletely-specified vector v_i is added to C.

After C is computed, it is reduced by considering every pair of cubes $c_1, c_2 \in C$. Suppose that c_1 covers c_2 (i.e., c_2 is specified to the same value as c_1 for every input where c_1 is specified). Suppose in addition that $S(c_2) = S(c_1)$. In this case, c_1 and c_2 result in the same subset of synchronized state variables, but c_1 has fewer specified inputs. It is sufficient in this case to keep c_1 and remove c_2 from C.

Several parameters are used in this process.

(1) If $S(c_i)$ consists of a small number of state variables, it may not be important to prevent c_i from appearing in a random input sequence T, since c_i only synchronizes a small number of state variables and may not prevent the detection of faults. The constant N_S is introduced to address this point. For a given value of N_S, a cube c_i is included in C only if the number of synchronized state variables under c_i is at least N_S, i.e., $|S(c_i)| \geq N_S$.

(2) If c_i has a large number of specified inputs, it is not likely to be a part of a random input sequence T often. In this case it may not be important to try and prevent the inclusion of c_i in T, since the likelihood of c_i being included in T is already low. A constant N_I is introduced to address this point. N_I is used as follows. Let the number of primary inputs of the circuit be n. If $N_I \geq n$, fully-specified input vectors v_i are used to produce input cubes. If $N_I < n$, the random input vectors v_i have exactly N_I specified values. Specified inputs and their values are selected randomly.

B. Modifying a Random Primary Input Sequence T

Given a set of input cubes C and a random primary input sequence T, the following process is used in [7] to exclude from T the input cubes included in C. We point out that the process is designed to be imperfect, i.e., it sometimes leaves in T some of the input cubes from C. This is important in order not to prevent the circuit from being synchronized, and not to prevent subsets of state variables from ever assuming certain combinations. Allowing a subset of state variables to be synchronized occasionally may be important for detecting certain faults.

Let t_u be the vector at time unit u of T. To eliminate the cubes in C from being included in t_u, the following process is applied. During a single pass of this process, the cubes in C are considered one at a time. For $c \in C$, if c covers t_u, new random values are selected under t_u for the inputs specified under c. A new pass is then started. Additional passes over the cubes in C are performed until either all the cubes in C are excluded from t_u, or a limit K on the number of passes is reached. The limit is reached if the cubes in C cannot all be excluded from t_u.

An input vector t_u is considered for modification with probability P; with probability $1-P$ t_u is not considered. A value of P close to one implies that most of the input vectors of T are modified to eliminate the appearance of cubes from C.

C. Parameter Selection

Several parameters determine the set of cubes C and the modified version of the random primary input sequence T. Three of the parameters, N_I, M and K, require large enough values to produce an effective set of cubes and a sufficiently modified random sequence, but otherwise do not have a significant effect on the results. In [7], these parameters were set to $N_I = 10$, $M = 10000$ and $K = 1000$. We use the same values here.

The remaining parameters are N_S and P. The experiments performed in [7] used all the possible values for N_S. A value of N_S is possible if there is an input cube that was found in the first phase and synchronizes exactly N_S state variables. For P, the values $P = 0/16, 1/16, \cdots, 16/16$ were used in [7]. The same input sequence T of length $L = 1000$ was modified using every combination of N_S and P, and fault simulation of T was carried out to compute the fault coverage of single stuck-at faults. The

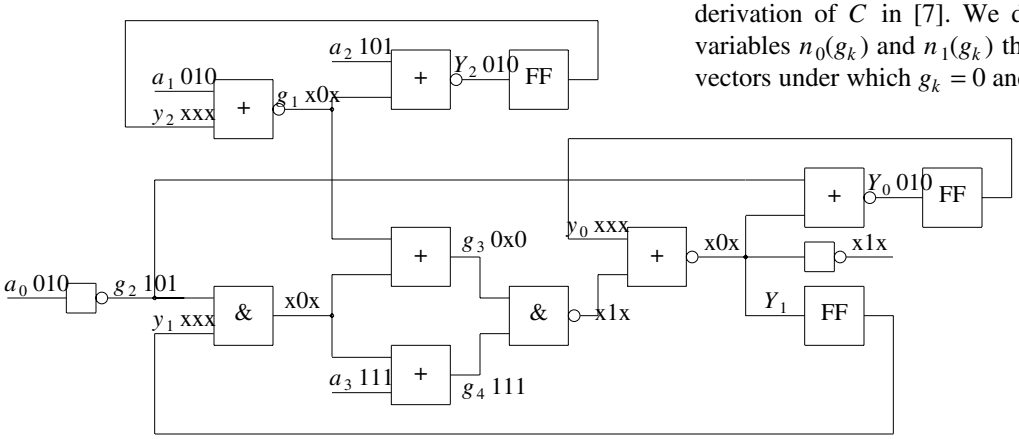

Fig. 1. ISCAS-89 benchmark circuit s27

results presented in [7] indicate that one of the smallest values of N_S with a value of P between 10/16 and 15/16 are most effective for all the benchmark circuits considered. We use similar values here. Based on the experience from [7], we consider increasing values of N_S, and decreasing values of P, where P = 15/16, 14/16, $\cdots$, 8/16. As in [7], we report results for the smallest value of N_S and the highest value of P that result in the highest fault coverage of all the values considered.

III. Using Additional Line Values

In this section we describe a procedure that considers the values of lines other than the state variables in order to define a set of input cubes C, which need to be avoided in a random input sequence. We denote by G the set of lines whose values will be considered.

For illustration we consider ISCAS-89 benchmark circuit $s27$ shown in Figure 1. The primary inputs are denoted by a_0, a_1, a_2 and a_3. The present-state variables are denoted by y_0, y_1 and y_2. The next-state variables are denoted by Y_0, Y_1 and Y_2. We show three primary input vectors in Figure 1, which are applied to the circuit each one separately when the circuit is in the all-unspecified state. These vectors will be used below.

A line g_k should be included in G if a random primary input sequence T is likely to set g_k to a specific value $w \in \{0,1\}$ repeatedly. This implies that it should be easy for T to set $g_k = w$, while T should be unlikely to set g_k to the opposite value w'. When T sets $g_k = w$ repeatedly, it may prevent the detection of faults whose activation or propagation requires $g_k = w'$. Including g_k in g will ensure that C will include input cubes that cause g_k to be set to w. Using C to modify T will then work to reduce the number of time units where $g_k = w$.

We measure the likelihood of a random primary input sequence to set $g_k = w$, for $w \in \{0,1\}$, by simulating the primary input vectors $\{v_i : 0 \le i < M\}$ used for the derivation of C in [7]. We define for every line g_k the variables $n_0(g_k)$ and $n_1(g_k)$ that hold the number of input vectors under which $g_k = 0$ and $g_k = 1$, respectively.

Initially, we set $n_0(g_k) = n_1(g_k) = 0$. For i = 0, 1, $\cdots$, $M-1$, we apply v_i to the primary inputs of the fault free circuit when the circuit is in the all-unspecified state. We compute the values throughout the circuit under v_i. For every line g_k, if g_k assumes the value 0 under v_i, we increment $n_0(g_k)$ by one; and if g_k assumes the value 1 under v_i, we increment $n_1(g_k)$ by one.

For illustration, we consider $s27$ with the three primary input vectors shown in Figure 1. Based on these three primary input vectors, and considering lines g_1, g_2, g_3 and g_4 marked in Figure 1, we obtain $n_0(g_1) = 1$, $n_1(g_1) = 0$, $n_0(g_2) = 1$, $n_1(g_2) = 2$, $n_0(g_3) = 2$, $n_1(g_3) = 0$, $n_0(g_4) = 0$ and $n_1(g_4) = 3$. After applying 20 random primary input vectors we obtain $n_0(g_1) = 8$, $n_1(g_1) = 0$, $n_0(g_2) = 6$, $n_1(g_2) = 14$, $n_0(g_3) = 3$, $n_1(g_3) = 0$, $n_0(g_4) = 3$ and $n_1(g_4) = 12$.

We note that if $n_w(g_k)$ is high for some $w \in \{0,1\}$, a random primary input sequence T is likely to set $g_k = w$ often. Our goal is to identify lines g_k such that $n_w(g_k)$ is high and $n_{w'}(g_k)$ is low, for some $w \in \{0,1\}$. Since we simulate a limited number of random vectors to compute $n_w(g_k)$, we consider $n_w(g_k) > 0$ an indication that g_k can be set to w by a random primary input sequence. To say that g_k is unlikely to be set to the value w, we require $n_w(g_k) = 0$. The lines of interest for inclusion in G are therefore the ones where either $n_0(g_k) = 0$ and $n_1(g_k) > 0$, or $n_0(g_k) > 0$ and $n_1(g_k) = 0$. We use the following definition of a variable $n_{01}(g_k)$ to capture these cases.

If $n_0(g_k) = 0$ and $n_1(g_k) > 0$, we set $n_{01}(g_k) = n_1(g_k)$. Otherwise, if $n_0(g_k) > 0$ and $n_1(g_k) = 0$, we set $n_{01}(g_k) = n_0(g_k)$. Otherwise, we set $n_{01}(g_k) = 0$. For illustration, we consider $s27$. Using the 20 random primary input vectors described above we obtain $n_{01}(g_1) = 8$, $n_{01}(g_2) = 0$, $n_{01}(g_3) = 3$ and $n_{01}(g_4) = 0$.

A line g_k can have $n_{01}(g_k) = 0$ for one of two reasons.

(1) The primary input vectors v_i, for $0 \le i < M$, assign both 0s and 1s to g_k. In this case, g_k is likely to be

assigned both 0 and 1 values by a random primary input sequence T, and there is no issue of repeatedly assigning the same value to g_k under T. We do not consider g_k further in this case.

(2) The primary input vectors v_i never assign a specified value to g_k. In this case, we will rely on the avoidance of repeated synchronization of the state variables to take care of assigning both possible values to g_k.

We are left with the lines for which $n_{01}(g_k) > 0$ as candidates for inclusion in G. For such a line g_k, a random primary input sequence T is likely to assign one value but not the other. Consequently, T may repeatedly assign the same value to g_k.

We define the importance of including G in g_k according to the value of $n_{01}(g_k)$. The higher the value, the higher the likelihood of assigning the same value to g_k repeatedly. Therefore, a line with a higher value of $n_{01}(g_k)$ is more important to include in G.

We use a constant $N01 > 0$ to determine the set G. Given a value for $N01$, we include in G every line g_k with $n_{01}(g_k) \geq N01$. In addition, we include in G all the next-state variables of the circuit. In the example of $s27$, with $N01 = 5$, g_4 would be included in G together with the next-state variables, while g_1, g_2 and g_3 would be excluded from G.

The possible values for $N01$ in a circuit with a set of lines $H = \{g_0, g_1, \cdots, g_{m-1}\}$ are all the values in the set $\{n_{01}(g_k) \neq 0 : 0 \leq k < m\}$. There is no need to consider a value for $N01$ that is not in this set for the following reason. Let $N\hat{0}1$ be a value not included in $\{n_{01}(g_k) \neq 0 : 0 \leq k < m\}$. Let $N01$ be the smallest value such that $N01 > N\hat{0}1$ and $N01 \in \{n_{01}(g_k) \neq 0 : 0 \leq k < m\}$. The set G computed for $N\hat{0}1$ is the same as the set computed for $N01$.

In addition to the values in $\{n_{01}(g_k) \neq 0 : 0 \leq k < m\}$, we consider $N01 = M+1$. Since we consider M primary input vectors for the computation of $n_{01}(g_k)$, it is guaranteed that $n_{01}(g_k) \leq M$ for $0 \leq k < m$. With $N01 = M+1$, only next-state variables are included in G. This yields the same results as in [7]. We use all the possible values of $N01$ in our experiments with the goal of identifying appropriate values for $N01$.

According to the definition used in [7], an input cube c synchronizes a subset of state variables $S(c)$ if applying c to the primary inputs, when the circuit is in the all-x state, results in the specification of the next-state variables in $S(c)$. We extend this definition as follows.

Given a set of lines G, an input cube c synchronizes a subset of lines $S(c) \subseteq G$ if applying c to the primary inputs, when the circuit is in the all-x state, results in the specification of the lines in $S(c)$.

With this definition, we can apply the procedures from [7] to find a set of input cubes C. We can then use the procedure from [7] to modify a random primary input sequence T so as to avoid the input cubes in C.

IV. Experimental Results

We computed a set of lines G as described in the previous section for every possible value of $N01$. We applied the procedure for computing a set of input cubes C using the set G obtained for every possible value of $N01$. For every value of $N01$ we used all the values of N_S and P to modify the same random primary input sequence T of length $L = 1000$.

In Table 1 we show all the possible values of $N01$ for every circuit for which the use of G improved the fault coverage of a random primary input sequence beyond the improvement achieved by using next-state variables only. With $M = 10000$, $N01 = 10001$ corresponds to the results from [7] where only next-state variables are considered. The value shown in bold is explained below.

TABLE 1
Values of $N01$ for Single Test Sequences

circuit	N01
s208	10001, 7101, 7089, **7088**, 7067, 7064, 7053, 7048, 7032, 7031, 5734, 5704, 4597, 4558
s386	10001, 8780, 7537, 7519, 7494, 7479, **7453**, 5086, 5043, 5033, 5017, 4983, 4974, 4967, 4957, 4950, 4914, 4418, 2521, 2506, 2476, 2392, 1282
s526	10001, 5099, 5051, **5015**, 4985, 4949
s641	10001, **6242**, 5602, 5593, 5544, 4752, 4743, 4715, 3903, 3895, 3870, 3861, 3830, 3829, 3799, 2849, 2848, 2784, 2765, 2764, 2669, 2657, 2650, 2642, $\cdots$
s1196	10001, 9970, 9957, 9951, 9920, 9913, 9909, 9907, 9869, 9864, 9846, 9825, 9812, 9809, 9791, 9770, 9766, 9753, 9732, 9714, 9688, 9641, 9628, 9610, 9597, 9594, 9566, 9554, 9515, 9471, 9404, 9402, 9337, 9331, 9296, 9020, 8950, 8930, 8883, 8843, 8831, 8750, 8730, 8707, **8686**, 8587, 8579, 8542, 8499, 8477, 8476, 8349, 8325, 8175, $\cdots$,
s1423	10001, 5103, 3034, 2999, 2997, 2994, 2993, **2991**, 2975, 2971, 2968, 2961, 2953, 2936, 2920, 2913, 2894, 2887, 2884, 2880, 2872, 901, 862, 854, 837, 835, 811, 245, 228, 211
s1488	10001, 8789, 8761, 8309, 7518, 7512, 7507, 7477, 6601, 6287, 6218, 6213, 5616, 5088, 5036, 5030, 5019, 5010, 5007, 5001, 4993, 4990, 4987, 4970, 4964, **4912**, 4438, 4412, 3800, 3778, 3748, 3734, 3725, 3695, $\cdots$
b04	10001, 8056, 6639, 4271, 4223, 4210, 4209, 4204, 4202, 4188, 4182, 4179, 4165, 4161, 4151, 4149, 4133, 4125, 4111, 4092, 4085, 2728, 1748, 1745, 1729, 1725, 1724, 1718, 1707, 1687, 1681, 726, 708, 660, 264, 105, **103**, 36, 22, 14, 11, 6, 4, 3, 1
b05	10001, 5029, **4971**, 4960
b07	10001, 5029, **4971**, 4960
b09	10001, **4971**, 4960
b10	10001, 8907, 8059, **8038**, 6664, 6553, 4223, 4204, 4182, 4179, 4165, 4151, 4133, 4125, 4111, 4085, 4076, 1709
b11	10001, 5030, 5013, 5010, **5007**, 5001, 4970, 4964, 4912
b14	10001, 1566, **1559**, 1555, 1550, 1549, 1548, 1547, 1545, 1540, 1529, 1521, 1520, 1519, 1507, 1506, 1505, 1501, 1500, 1498, 1496, 1493, 1487, 1482, $\cdots$,

TABLE 2
Results Using Single Test Sequences

	best mod random						rand	determ
circuit	N01	G-sv	NS	P	det	f.c.	f.c.	f.c.
s208	10001	0	4	13/16	136	63.26	36.74	63.72
s208	7088	3	3	13/16	137	*63.72	36.74	63.72
s298	10001	0	8	14/16	265	*86.04	60.39	86.04
s344	10001	0	7	12/16	329	*96.20	95.91	96.20
s382	10001	0	21	15/16	347	86.97	12.28	91.23
s386	10001	0	4	10/16	305	79.43	55.73	81.77
s386	7453	35	13	12/16	310	80.73	55.73	81.77
s400	10001	0	21	15/16	364	86.46	12.11	90.26
s420	10001	0	4	14/16	179	*41.63	26.98	41.63
s526	10001	0	15	15/16	389	70.09	8.65	81.80
s526	5015	19	2	15/16	419	75.50	8.65	81.80
s641	10001	0	1	11/16	403	86.30	83.51	86.51
s641	6242	1	1	13/16	404	*86.51	83.51	86.51
s1196	10001	0	14	15/16	1095	88.16	76.81	99.76
s1196	8686	70	78	9/16	1159	93.32	76.81	99.76
s1423	10001	0	56	15/16	1193	78.75	41.45	93.33
s1423	2991	13	9	15/16	1262	83.30	41.45	93.33
s1488	10001	0	6	14/16	1367	91.99	57.87	97.17
s1488	4912	142	54	15/16	1417	95.36	57.87	97.17
s5378	10001	0	14	14/16	3375	73.32	63.42	79.06
b03	10001	0	30	14/16	334	*73.89	64.60	73.89
b04	10001	0	66	14/16	1156	85.88	74.52	86.78
b04	103	175	83	14/16	1165	86.55	74.52	86.78
b05	10001	0	34	15/16	1073	59.09	30.73	59.25
b05	4971	10	2	15/16	1074	59.14	30.73	59.25
b07	10001	0	51	15/16	835	70.58	4.73	70.75
b07	4971	52	5	15/16	836	70.67	4.73	70.75
b08	10001	0	21	15/16	445	91.00	5.93	94.68
b09	10001	0	28	15/16	230	54.76	22.62	81.19
b09	4971	1	1	13/16	295	70.24	22.62	81.19
b10	10001	0	17	14/16	454	88.67	66.02	91.21
b10	8038	11	11	13/16	466	91.02	66.02	91.21
b11	10001	0	30	15/16	914	83.93	19.01	92.19
b11	5007	5	5	15/16	938	86.13	19.01	92.19
b14	10001	0	247	15/16	7008	70.21	44.64	88.12
b14	1559	27	27	15/16	7128	71.42	44.64	88.12

The results of modifying a random primary input sequence are shown in Table 2. We show up to two rows for every circuit. The first row corresponds to the results from [7], which are obtained here with $N01 = 10001$. This value of $N01$ does not allow any lines other than next-state variables to be included in G. The second row corresponds to the highest value of $N01$ for which the highest fault coverage was obtained. The second row is omitted when $N01 = 10001$ yielded the highest fault coverage. In this case, inclusion of lines that are not next-state variables in G does not improve the fault coverage of T. If an increased fault coverage is obtained with $N01 < 10001$, the circuit is included in Table 1 and the value of $N01$ is shown in bold in Table 1.

Every row of Table 2 is organized as follows. Under column *best mod random* we show the value of $N01$ followed by the number of lines in G that are not next-state variables. For the selected value of $N01$, we show the smallest value of N_S that resulted in the highest fault coverage. For this value of N_S we show the largest value of P that resulted in the highest fault coverage. Under subcolumn *det* we show the number of detected faults obtained for N_{01}, N_S and P, followed by the corresponding fault coverage. The number of detected faults and the fault coverage under the modified random input sequence are given for a single sequence T of length $L = 1000$. For comparison, under column *rand f.c.* we show the fault coverage achieved for the random primary input sequence T without modification. Under column *determ f.c.* we show the fault coverage reported for deterministic test generation procedures. We place an asterisk before the fault coverage of the modified random input sequence when it is equal to the deterministic fault coverage.

From Table 2 it can be seen that including lines that are not state variables in G improves the fault coverage of T after it is modified. In several cases, the deterministic fault coverage is reached due to the use of G.

From Table 1, the value of $N01$ that results in the highest fault coverage is typically close to the maximum value of $N01$. When the highest fault coverage is obtained for a low value of $N01$, there is typically a high value of $N01$ that yields almost the same fault coverage. This indicates that it is sufficient to include in G a small number of lines, which have the highest values of $n_{01}(g_k)$, in addition to the state variables.

In the experiment reported in Table 2 we considered the best fault coverage achievable with a single modified random primary input sequence. It is also possible to consider the cumulative fault coverage of several modified random primary input sequences. Different modified test sequences can be obtained by varying $N01$, N_S, and P. Based on Tables 1 and 2, we use the 10 highest values of $N01$ for every circuit. Based on [7], we use the three smallest values of N_S, and $P = 15/16, 14/16, \cdots, 8/16$. In addition, it is possible to use different random primary input sequences as starting points for the modification.

In the experiment reported next, we consider random input sequences $T_1, T_2, \cdots$ of length $L = 1000$. For every input sequence T_i, we modify the sequence using every combination of $N01$, N_S and P separately. For the sequence obtained from T_i based on $N01$, N_S and P, we perform fault simulation with fault dropping.

We continue to consider input sequences until no new fault is detected based on the last constant number of sequences. We use 10 as the constant limit.

In Table 3 we show the values of $N01$ that were used for modifying at least one test sequence that increased the fault coverage. We show up to 10 of the largest possible values for every circuit, which were used during the modification process. The useful values are marked in bold.

TABLE 3
Values of $N01$ for Multiple Test Sequences

circuit	N01
s382	**10001**, **5099**, 5051, **5015**, **4985**, 4901
s386	**10001**, 8780, 7537, 7519, 7494, 7479, 7453, 5086, 5043, 5033
s400	**10001**, **5099**, 5051, **5015**, **4985**, 4901
s526	**10001**, 5099, **5051**, **5015**, **4985**, **4949**
s1196	**10001**, **9970**, 9957, 9951, **9920**, **9913**, **9909**, **9907**, **9869**, **9864**
s1423	**10001**, 5103, **3034**, **2999**, **2997**, 2994, 2993, **2991**, 2975, 2971
s1488	**10001**, **8789**, **8761**, 8309, **7518**, **7512**, 7507, 7477, 6601, 6287
s5378	**10001**, **7375**, **7363**, **7354**, 7351, 7344, 7338, 7329, 7320, **7311**
b04	**10001**, 8056, **6639**, 4271, **4223**, 4210, 4209, 4204, 4202, 4188
b05	**10001**, 5029, **4971**, 4960
b07	**10001**, 5029, 4971, 4960
b08	**10001**, **5101**, **5093**, **5034**, 5029, 5027, 5008, 5006, 4966, 4956
b09	**10001**, **4971**, 4960
b10	**10001**, **8907**, **8059**, 8038, 6664, 6553, 4223, 4204, 4182, 4179
b11	**10001**, **5030**, **5013**, **5010**, **5007**, **5001**, **4970**, 4964, **4912**
b14	**10001**, **1566**, **1559**, **1555**, **1550**, 1549, **1548**, **1547**, **1545**, **1540**

The fault coverage achieved by using multiple modified random primary input sequences as described above is shown in Table 4. We consider circuits for which the fault coverage achieved in Table 2 is lower than the deterministic fault coverage. Under column *mult mod* we show the following information. Under subcolumn *seq* we show the number of modified test sequences that were effective in improving the fault coverage. Under column $f.c.$ we show the fault coverage achieved by using all the test sequences. In the last column we repeat the deterministic fault coverage.

TABLE 4
Results Using Multiple Test Sequences

			mult mod		determ
circuit	inp	sv	seq	f.c.	f.c.
s382	3	21	5	90.98	91.23
s386	7	6	6	*81.77	81.77
s400	3	21	5	90.02	90.26
s526	3	21	15	81.62	81.80
s1196	14	18	45	*99.76	99.76
s1423	17	74	44	89.77	93.33
s1488	8	6	21	*97.17	97.17
s5378	35	179	47	76.52	79.06
b04	12	66	8	*86.78	86.78
b05	2	34	2	*59.25	59.25
b07	2	51	3	*70.75	70.75
b08	10	21	7	*94.68	94.68
b09	2	28	9	*81.19	81.19
b10	12	17	8	*91.21	91.21
b11	8	30	14	91.55	92.19
b14	33	247	84	80.56	88.12

From Table 4 it can be seen that the deterministic fault coverage is achieved for several additional circuits considered, and the fault coverage was improved for the other circuits.

From Table 3 it can be seen that except for two circuits, $s386$ and $b07$, values of $N01$ smaller than 10001 were needed to include lines other than next-state variables in G and thus achieve the final fault coverage.

V. Concluding Remarks

We described an improved procedure for increasing the fault coverage of a random primary input sequence. An earlier procedure modified a random primary input sequence so as to avoid repeated synchronization of state variables. The proposed procedure considered, in addition to state variables, other lines that may be repeatedly set to the same values by a random primary input sequence. When such a line is allowed to assume the same value repeatedly, faults that require the opposite value for activation or propagation are prevented from being detected. We described a procedure for selecting lines whose values are important for determining the fault coverage of a random primary input sequence. The lines were selected based on simulation of random primary input vectors. A line that assumed only one value a large number of times was considered as a target when modifying a random primary input sequence. We presented experimental results to demonstrate the improvements in fault coverage of random primary input sequences when the values of the selected lines were considered.

References

[1] R. Marlett, "An Effective Test Generation System for Sequential Circuits", in Proc. Design Autom. Conf. 1986, pp. 250-256.

[2] W.-T. Cheng and T. J. Chakraborty, "Gentest: An Automatic Test Generation System for Sequential Circuits", IEEE Computer, April 1989, pp. 43-49.

[3] T. P. Kelsey and K. K. Saluja, "Fast Test Generation for Sequential Circuits", in Proc. Intl. Conf. on Computer-Aided Design, 1989, pp. 354-357.

[4] T. Niermann and J. H. Patel, "HITEC: A Test Generation Package for Sequential Circuits", in Proc. European Design Autom. Conf., 1991, pp. 214-218.

[5] X. Lin, I. Pomeranz and S. M. Reddy, "Techniques for Improving the Efficiency of Sequential Circuit Test Generation", in Proc. Intl. Conf. on Computer-Aided Design, 1999, pp. 147-151.

[6] P. C. Maxwell, R. C. Aitken, K. R. Kollitz and A. C. Brown, "IDDQ and AC Scan: The War Against Unmodelled Defects", in Proc. Intl. Test Conf., Oct. 1996, pp. 250-258.

[7] I. Pomeranz and S. M. Reddy, "Primary Input Vectors to Eliminate from Random Test Sequences for Synchronous Sequential Circuits", in Informal Digest of Papers, European Test Symp., May 2007.