AN EFFICIENT TEST GENERATION ALGORITHM BASED ON SEARCH STATE DOMINANCE

Takayuki Fujino
Hideo Fujiwara

Reprinted from PROCEEDINGS OF THE FTCS-22 — THE TWENTY-SECOND INTERNATIONAL SYMPOSIUM ON FAULT-TOLERANT COMPUTING, Boston, Massachusetts, July 8-10, 1992

An Efficient Test Generation Algorithm Based on Search State Dominance

Takayuki Fujino and Hideo Fujiwara

Department of Computer Science
Meiji University
1-1-1 Higashimita, Tama-Ku, Kawasaki 214, Japan

Abstract:

Giraldi and Bushunell proposed a new test generation method, called the EST (Equivalent STate hashing) algorithm, which can reduce the search space for test generation by using Binary Decision Diagram fragments to detect previously encountered search states. In this paper, we extend the concept of search state equivalence to that of search state dominance, and propose a new extended method, DST (Dominant STate hashing) algorithm, based on the search state dominance. The DST algorithm can prune the search space more effectively than the EST algorithm.

Key words:

Combinational Logic Circuits, Decision Trees, Search State Dominance, Search Space Pruning, Test Pattern Generation

I. Introduction

Many test generation algorithms for combinational circuits have been proposed over the years [1, 3-12]. Among those algorithms, the D-algorithm [3] was the first *complete* algorithm that can generate a test-pattern for any logical fault if such a test-pattern exists and enough computing time is given. The second significant progress in accelerating algorithms was achieved by PODEM [4] and FAN [5, 6]. However, the computational resources required for test generation were still immense, i.e., there still remained some aborted faults in the ISCAS'85 benchmarks [2] due to the limited computing time. After that, some approaches and improvements [9-12] have been proposed and succeeded in handling all faults in the ISCAS'85 benchmarks by either generating a test-pattern or a redundancy proof. Among those approaches, Giraldi and Bushnell [12] proposed a new test generation method, called the EST (Equivalent STate hashing) algorithm, which can reduce the search space for test generation by using Binary Decision Diagram fragments to detect previously encountered search states. The EST algorithm has a characteristic feature such that it is orthognal to all existing test generation algorithms, so it can be used to accelerate any test-pattern generator.

In this paper, we extend the concept of search state equivalence to that of search state dominance, and propose a new extended method, DST (Dominant STate hashing) algorithm, based on the search state dominance. The DST algorithm can prune the search space more effectively than the EST algorithm. First, we introduce the EST algorithm of Giraldi and Bushnell and some concepts; search state, evaluation frontier, and search state equivalence. Next, we extend the concept of search state equivalence to search state dominance, and then present some theorems and the DST algorithm. We illustrate the benefits of DST through examples of decision trees.

II. The EST Algorithm

The problem of generating a test-pattern for a given fault can be viewed as a finite space search problem of finding a point in the search space that corresponds to a test-pattern. Test generation algorithms like PODEM [4] make a series of primary input (PI) assignments for fault

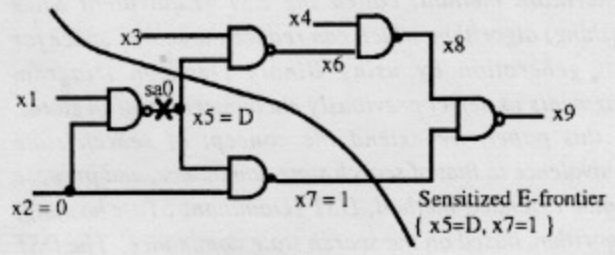
0731-3071/92 $03.00 © 1992 IEEE

sensitization and propagation in the circuit. When a fault is sensitized, a D-frontier appears in the circuit, where a *D-frontier* is defined as a set of gates with unspecified outputs and some input signal set to D or $\overline{D}$. Further PI assignments for advancing the D-frontier to a primary output (PO) of the circuit find a test-pattern. During this process, backtracking occurs when either the D-frontier disappears or the fault site is not sensitized. In this way, test generation algorithms usually build a decision tree and apply a backtracking search procedure. Each node or step in the decision tree corresponds to a partial PI value assignment. Implication or five-valued (0, 1, X, D, $\overline{D}$) logic simulation for the partial PI assignment forms a decomposition in the circuit. Since each node in the decision tree corresponds to a search state, the search state is defined as the logic circuit decomposition derived from the PI assignment corresponding to the decision step. Figure 1a (Figures 1b and 1c) show decompositions for sensitized (unsensitized) faults.

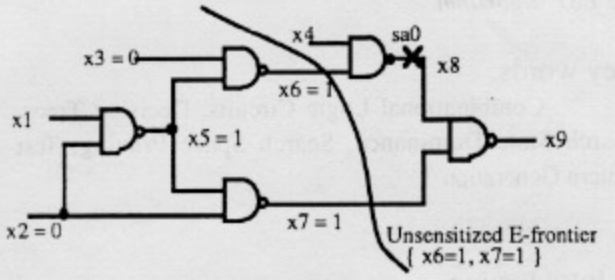
Giraldi and Bushnell [12] introduced *evaluation frontier (E-frontier)* to represent a search state more efficiently. The E-frontier represents a complete circuit cut-set labeling and uniquely identifies a circuit decomposition. In a five-valued (0, 1, X, D, $\overline{D}$) test generation algorithm, an E-frontier consists of all internal nets labeled with values other than X (unassigned) that are connected to the circuit PO's by a path of gates with unassigned values (an X-path). In Figure 1(a), the PI assignment is {x2=0}, and the E-frontier is {x5=D, x7=1}. In Figures 1(b) and 1(c), those PI assignments are {x2=0, x3=0} and {x1=1, x2=1}, respectively, and have the same (equivalent) E-frontier {x6=1, x7=1}. Subsequent operations on such equivalent circuit decompositions produce equivalent results, and hence we define *search state equivalence* as follows: search states are called to be equivalent if their E-frontiers are equivalent.

Giraldi and Bushnell proposed an approach to early identification of search path termination conditions by using E-frontiers, i.e., pruning search space. Consider the incomplete decision tree of Figure 2 representing the search space for fault f. Node letters represent search states at each decision step. E-frontiers are computed at each node. Search starts at the root node A and proceeds in depth-first search order up to node G with implication resulting in search state C'. Suppose that the E-frontier of C' is equivalent to that of C. Then we can back up (backtrack) without exploring implications from C' since search state C equivalent to C' is known to be inconsistent.

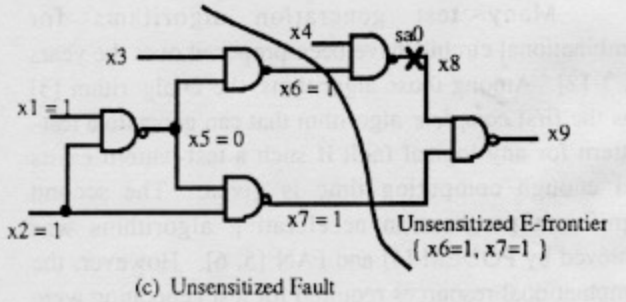
Consider another early search termination example. Suppose that a test-pattern for fault f is generated as shown in Figure 2. Suppose that we start searching a test-pattern for another fault g. When x2 is set to 0 in search state L, search state H' is created and found to be equivalent to node H for fault f. If the fault has been sensitized, i.e., the E-frontier contains the value D or $\overline{D}$, then all subsequent PI assignments for fault g will be identical to those previously made for fault f. Since node H for fault f is on a path to a test-pattern, the unassigned PI's for fault g are set to those corresponding PI assignments for fault f and search terminates immediately with a test-pattern for fault g.



(a) Sensitized Fault



(b) Unsensitized Fault



(c) Unsensitized Fault

Figure 1. Decompositions for Sensitized and Unsensitized Faults

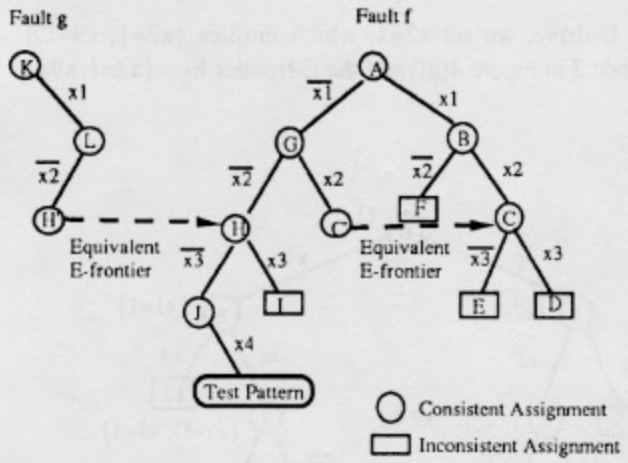


Figure 2. Equivalent States in Current and Prior Faults

III. Search State Dominance

In this section, we extend the concept of search state equivalence to that of search state dominance and present theorems for search path termination.

An E-frontier E can be represented by a set of pairs of net N and its value v, i.e., $E = \{(N_1, v_1), (N_2, v_2), ..., (N_k, v_k)\}$ or $E = \{ N_1=v_1, N_2=v_2, ... , N_k=v_k\}$. Let E_i and E_j be E-frontiers. We say that E_i *dominates* E_j if

(1) any pair (N,v) in E_i is included in E_j

(i.e., $E_i \subseteq E_j$), and

(2) any pair (N, v) in E_j such that v = D or $\overline{D}$ is included in E_i (i.e., both E_i and E_j contain the same D-frontier).

For example, consider four E-frontiers, E_1 = { x1=0, x2=1 }, E_2 = { x1=0, x2=1, x3=1, x4=0 }, E_3 = {x1=0, x2=1, x3=1, x4=$\overline{D}$ }, and E_4 = { x1=0, x3=1, x4= $\overline{D}$ }. E_1 dominates E_2 since $E_1 \subseteq E_2$. However, E_1 does not dominate E_3 since D-frontiers of E_1 and E_3 are not the same. On the other hand, E_4 dominates E_3 since $E_4 \subseteq E_3$ and both E_4 and E_3 contain the same D-frontier (x4=$\overline{D}$).

An E-frontier E is said to be *sensitized* if it contains value D or $\overline{D}$, i.e., the D-frontier of E is not empty. An E-frontier E is said to *have a solution* if there is a path from the node of the E-frontier to a node of a test-pattern in the decision tree.

In the following, we present two theorems for early identification of search path termination. The first is the theorem in case of searching a test-pattern for the same target fault.

Theorem 1: Let E_i and E_j be E-frontiers for the same target fault f. If E_i dominates E_j and E_i has no solution, then E_j has no solution.

Proof : Here we shall consider the PODEM algorithm [4] as the base test generation algorithm for the DST algorithm.

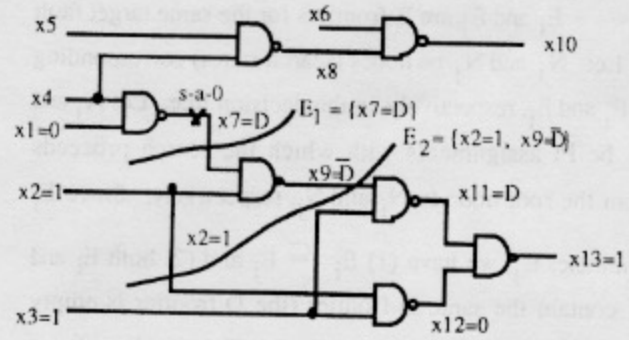
E_i and E_j are E-frontiers for the same target fault f. Let N_i and N_j be nodes (search states) corresponding to E_i and E_j, respectively, in the decision tree. Let A_i and A_j be PI assignments with which the search proceeds from the root node to N_i and N_j, respectively. Since E_i dominates E_j, we have (1) $E_i \subseteq E_j$ and (2) both E_i and E_j contain the same D-frontier (the D-frontier is empty when E_i and E_j are unsensitized). This implies that a node N_j' whose E-frontier is E_j can be reached from node N_i corresponding to E_i by assigning some values on unassigned PI's. Let A_{ij} be those corresponding PI assignments that transfer the search state from E_i to E_j. Hence, node N_j' can be reached from the root node by PI assignments A_i followed by A_{ij}.

Suppose that E_i has no solution for fault f but E_j has a solution for the same fault f. Since E_j has a solution for fault f, node N_j corresponding to E_j in the decision tree is on a path to a test-pattern for fault f. Let A_{jt} be those corresponding PI assignments that transfer from node N_j to the node where a test-pattern has been generated. The resulting test-pattern is the PI assignments A_j followed by A_{jt}. Since N_j' has the same

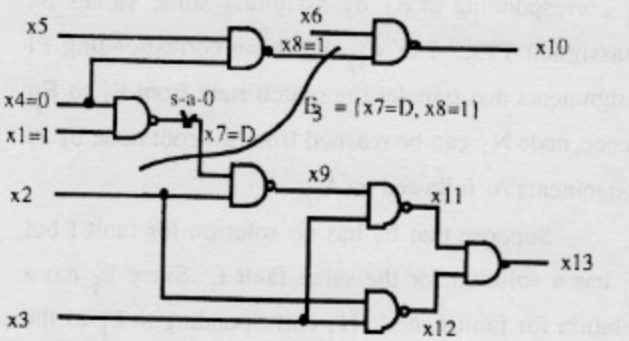
E-frontier as N_j, node N_j' is also on a path to a test-pattern for fault f. Furthermore, since N_j' can be reached from N_i, node N_i is also on a path to a test-pattern for fault f. That is, the PI assignments A_i followed by A_{ij} and A_{jt} can be the test-pattern for fault f. This contradicts that E_i has no solution for fault f. Hence, if E_i dominates E_j and E_i has no solution, then E_j has no solution.

Q.E.D.

Example 1: Let us try to generate a test-pattern for a fault x7 s-a-0 in the circuit of Figure 3(a). Let us first consider a conventional approach to searching a test-pattern for the fault. The decision tree is shown in Figure 4(a). First we must set x7=1 to activate the fault x7 s-a-0. To justify x7=1 we first try x1=0. This implies x7=D.



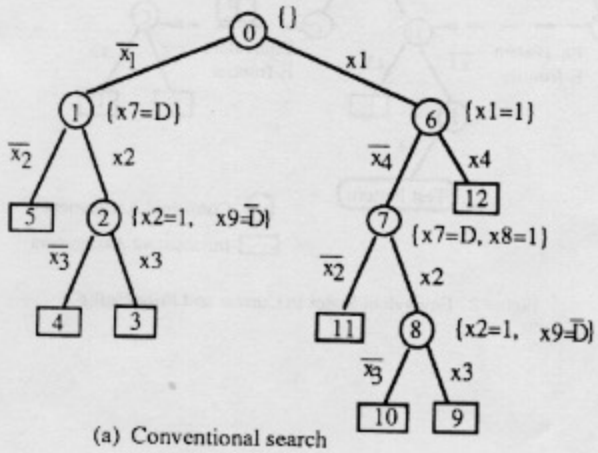
(a) Search state of node 3 in Figure 4



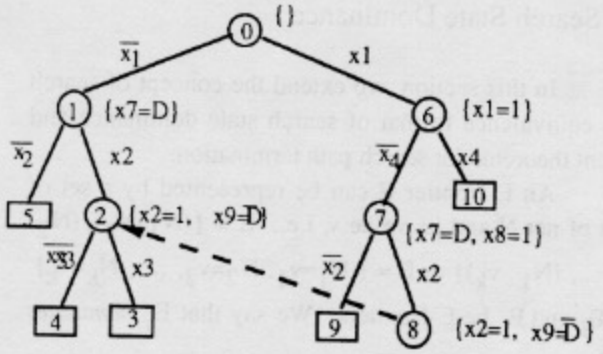
(b) Search state of node 7 in Figure 4

Figure 3. E-frontiers in Test Generation

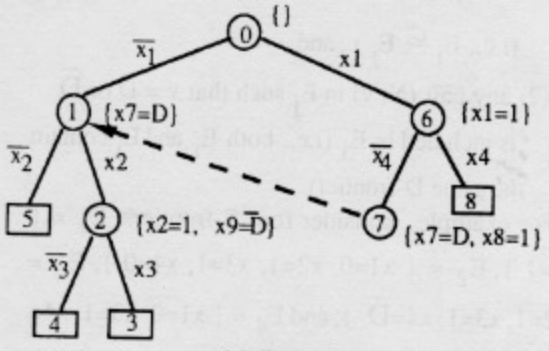
This state corresponds to node 1 in Figure 4(a) and the E-frontier is E_1={x7=D}. To propagate the error or D-drive, we set x2=1, which implies {x2=1, x9=$\bar{D}$} (node 2 in Figure 4(a)) and the E-frontier E_2 ={x2=1, x9=



(a) Conventional search



(b) Equivalence



(c) Dominance

Figure 4. Comparison of Equivalence and Dominance

$\overline{D}$]. To D-drive furher, we try to set x3=1. However, this leads x11=D, x12=0 and x13=1, i.e., empty D-frontier, which implies an inconsistency (node 3 in Figure 4(a)). Hence we must backtrack to node 2 in the decision tree in Figure 4(a) and reverse the value, i.e., x3=0 (node 4 in Figure 4(a)). In this way, the test-pattern generation proceeds exhaustively until it reaches the final node 12 of the decision tree in Figure 4(a) and fails to generate a test-pattern for the fault x7 s-a-0. The fault x7 s-a-0 is undetectable or redundant.

Next let us consider the EST algorithm [12]. At node 8 of the decision tree in Figure 4(a) the E-frontier E_2 ={x2=1, x9=$\overline{D}$} is equivalent to the E-frontier at node 2. The E-frontier also contains a D-frontier, i.e., the fault is sensitized and propagating at both nodes 2 and 8. Since the E-frontier of node 2 has no solution and both E-frontiers of nodes 2 and 8 are the same, the E-frontier of node 8 also has no solution, and hence we can backtrack without further search from node 8. EST can avoid to waste the time exploring implications from node 8. This process is illustrated in Figure 4(b).

If we look at node 7 in the decision tree of Figure 4(a), we find out that the E-frontier of node 7, {x7=D, x8=1}, is dominated by the E-frontier of node 1, {x7=D}. Since we know that there is no solution under node 1 in the decision tree, the E-frontier of node 1 has no solution. Hence, from Theorem 1 we can see that the E-frontier of node 7 also has no solution, and we can backtrack from node 7 to node 6 without further search under node 7. This is illustrated in Figure 4(c). By using the dominance relation of E-frontiers, we can terminate unnecessary searching earlier than the EST algorithm.

Next, we present a theorem for search path termination based on dominant search states in current and prior target faults.

Theorem 2: Let E_i and E_j be E-frontiers which are sensitized for target faults f_i and f_j, respectively.

(a) If E_i dominates E_j and E_j has a solution, then E_i has a solution.

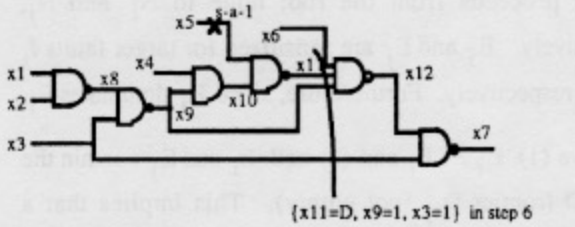
(b) If E_i dominates E_j and E_i has no solution, then E_j has no solution.

Proof: By the law of contraposition, (b) can be implied from (a). So, we shall prove (a) in the following.

Let N_i and N_j be nodes (search states) corresponding to E_i and E_j, respectively, in the decision tree. Let A_i and A_j be PI assignments with which the search proceeds from the root node to N_i and N_j, respectively. E_i and E_j are sensitized for target faults f_i and f_j, respectively. Furthermore, since E_i dominates E_j, we have (1) $E_i \subseteq E_j$ and (2) both E_i and E_j contain the same D-frontier D_{ij} (not empty). This implies that a node N_j' whose E-frontier is E_j can be reached from node N_i corresponding to E_i by assigning some values on unassigned PI's. Let A_{ij} be those corresponding PI assignments that transfer the search state from E_i to E_j. Hence, node N_j' can be reached from the root node by PI assignments A_i followed by A_{ij}. Since E_i and E_j have the same D-frontier D_{ij}, nodes N_j and N_j' have the same D-frontier D_{ij}. The D-frontiers of N_i and N_j' are sensitized for fault f_i, and the D-frontier of N_j is sensitized for fault f_j.

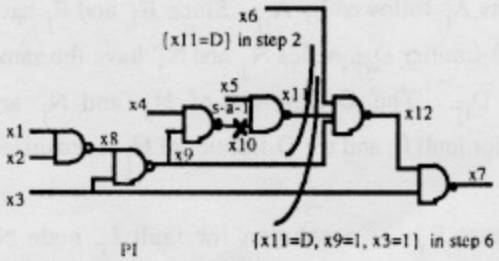
Since E_j has a solution for fault f_j, node N_j corresponding to E_j is on a path to a test-pattern for fault f_j. Let A_{jt} be those corresponding PI assignments that transfer from node N_j to the node where a test-pattern for fault f_j has been generated. The resulting test-pattern is the PI assignments A_j followed by A_{jt}. The PI assignments A_{jt} can propagate at least one of the sensitized values (for fault f_j) in the D-frontier D_{ij} of E_j to at least one primary output (PO). Since N_j' has the same E-frontier as N_j, at least one of the sensitized valued (for fault f_i) in the D-frontier of N_j' can be propagated to at least one PO by PI assignments A_{jt}. Therefore, node N_i is also on a path to a test-pattern for fault f_i. That is, the PI assignments A_i followed by A_{ij} and A_{jt} can be the test-pattern for fault f_i. Hence E_i has a solution for fault f_i. Q.E.D.

Example 2: Let us consider two faults, x5 s-a-1 and x10 s-a-1, in the circuit of Figure 5. First we consider to generate a test-pattern for the fault x5 s-a-1. In Figure 5(a), the test-pattern generation process for fault



Step	PI Assignment	E-frontier	
1	x5=0	{x5=$\overline{D}$}	
2	x3=0	{x7=1}	← Backtrack
3	x3=1	{x5=$\overline{D}$, x3=1}	
4	x1=1	{x5=$\overline{D}$, x3=1, x1=1}	
5	x2=1	{x5=$\overline{D}$, x9=1, x3=1}	
6	x4=1	{x11=D, x9=1, x3=1}	← Test-pattern generated
7	x6=1	{x7=D}	

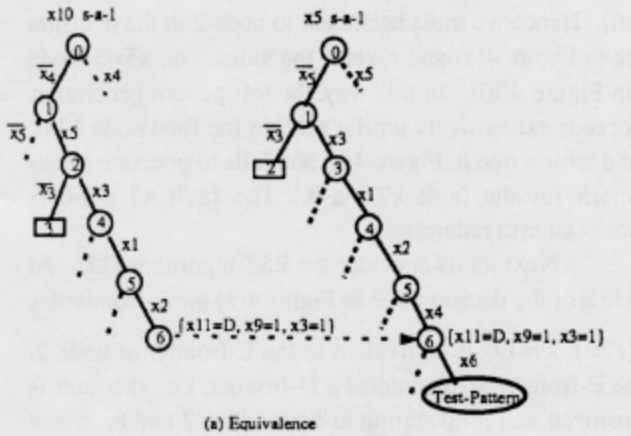
(a) x5 s-a-1 fault



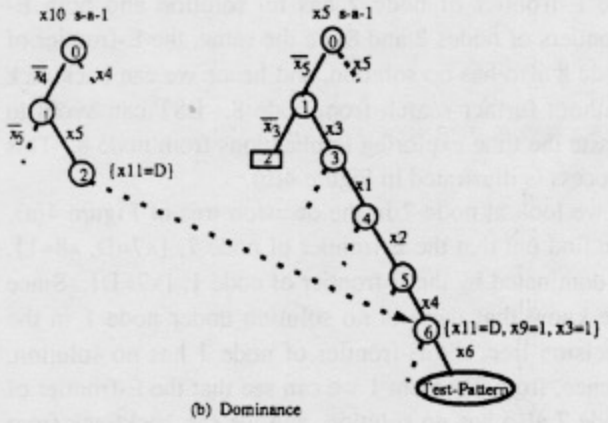
Step	PI Assignment	E-frontier	
1	x4=0	{x10=$\overline{D}$}	
2	x5=1	{x11=D}	← Dominates step 6 in (a)
3	x3=0	{x7=1}	← Backtrack
4	x3=1	{x11=D, x3=1,}	
5	x1=1	{x11=D x3=1, x1=1}	← Equivalent to step 6 in (a)
6	x2=1	{x11=D, x9=1, x3=1}	← Test-pattern generated
7	x6=1	{x7=D}	

(b) x10 s-a-1 fault

Figure 5. Test generation for (a) x5 s-a-1 and (b) x10 s-a-1 faults



(a) Equivalence



(b) Dominance

Figure 6. Comparison of Equivalence and Dominance

x5 s-a-1 is illustrated. The test-pattern is (x1, x2, x3, x4, x5, x6)=(1, 1, 1, 1, 0, 1).

Consider the subsequent search for fault x10 s-a-1 in Figure 5(b). The test-pattern generation proceeds as indicated in Figure 5(b). The corresponding decision tree is shown in Figure 6(a). According to the EST algorithm, only the equivalence relation among E-frontiers is checked. Hence, in this example, at step 6 in Figure 5(b) the computed E-frontier, {x11=D, x9=1, x3=1}, is found to be identical to the E-frontier in step 6 of the decision tree for the fault x5 s-a-1 (Figures 5(a)). The E-frontier contains D-frontier. Hence, all subsequent PI assignments for fault x10 s-a-1 is identical to those previously made for fault x5 s-a-1. Since node 6 for fault x5 s-a-1 is on a path to a test-pattern (Figure 6(a)), the unassigned PI's for fault x10 s-a-1 are set to those corresponding PI assignments for fault x5 s-a-1, i.e., x6=1, and search terminates immediately with a test-pattern for fault x10 s-a-1, (x1, x2, x3, x4, x5, x6) = (1, 1, 1, 0, 1, 1).

If we look at step 2 in Figure 5(b) or node 2 in the decision tree of Figure 6(b), we find out that the E-frontier in step 2, {x11=D}, dominates the E-frontier, {x7=D}, in step 6 of decision tree for the fault x5 s-a-1 (Figure 5(a)). In Figure 6(b), node 6 for fault x5 s-a-1 is on a path to a test-pattern. Therefore, from Theorem 2 we can see that fault x10 s-a-1 also has a test-pattern. The test-pattern is immediately obtained by setting the unassigned PI's for fault x10 s-a-1 to those corresponding PI assignments for fault x5 s-a-1, i.e., (x1, x2, x3, x6) = (1, 1, 1, 1). The PI assignment at node 2 in the decision tree for fault x10 s-a-1 is (x4, x5) = (0, 1). Hence the test-pattern for x10 s-a-1 is (x1, x2, x3, x4, x5, x6) = (1,

1, 1, 0, 1, 1). Figure 6 shows the comparison of two decision trees based on search state equivalence and dominance. We can see that if we use the dominance relation of E-frontiers, we can reduce the size of search space and hence the time to search the space more effectively than the EST algorithm.

IV. The DST Algorithm

In this section, we present the DST (Dominant STate hashing) algorithm based on the search state dominance which is an extension of the EST algorithm [12]. In the same way as the EST algorithm, the DST algorithm can be added as a sub-algorithm to any test-pattern generation algorithm, i.e., it is orthogonal to the operations of the base test-pattern generation algorithm and works with any test-pattern generation.

Figure 7 shows the DST algorithm. A hash table is used to determine search state dominance. Each E-frontier is stored in the hash table with the data associated with the E-frontier, which includes the target fault, the solution flag (solution / no solution) and the test-pattern (if exists). After implication in the base test generation algorithm, the DST algorithm starts and ends in one of the following three cases:

(1) exit with a test-pattern,

(2) exit to backtrack, or

(3) exit to continue the base test generation algorithm normally.

After starting the DST algorithm, each new E-frontier is computed and hashed into the hash table. All E-frontiers that dominate or are dominated by the current E-frontier are pushed on a stack. If the stack is empty, the base test generation algorithm continues normally. While the stack is not empty, each stack entry is examined. Let Ec be the current E-frontier and let Es be the E-frontier of the stack entry.

In case that Es is for the same fault as Ec, we further examine Es as follows: If Es dominates Ec and Es has no solution, the algorithm exits to backtrack (Theorem 1). Otherwise, the algorithm pops the stack and continues the stack loop. In case that Es is for a different fault from Ec, we further examine as follows: If neither Ec nor Es is sensitized, the base test generation algorithm continues normally. If Ec and Es are both sensitized and if Ec dominates Es and Es has a solution, the test-pattern is formed and the unstacking loop is exited with a test-pattern for the current fault (Theorem 2(a)). If Ec and Es are both sensitized and if Es dominates Ec and Es has no solution, the algorithm exits to backtrack (Theorem 2(b)).

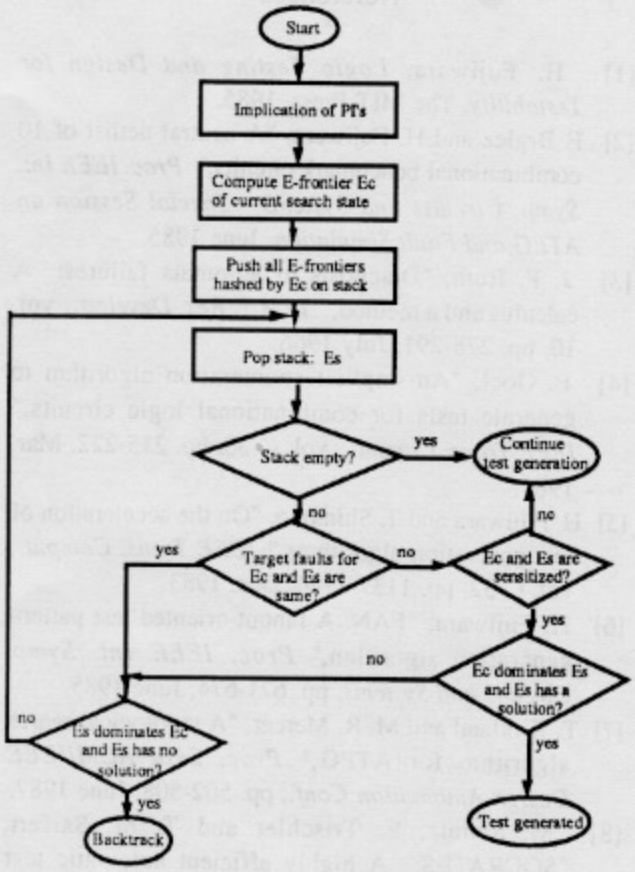


Figure 7. DST Algorithm

V. Conclusions

We have presented a new test-pattern generation algorithm (called DST) based on a new concept called search state dominance. Search state dominance is an extended concept of search state equivalence introduced by Giraldi and Bushnell [12]. We have presented some techniques which can prune search space and accelerate a test-pattern generation algorithm. Some theorems have been shown to guarantee the effectiveness of the search space pruning. We have finally presented the DST algorithm which can be added to any test-pattern generation algorithm as a sub-algorithm. The DST algorithm has the high possibility of pruning search

space more effectively than the EST algorithm of Giraldi and Bushnell [12]. We are currently implementing the DST algorithm with PODEM [4] and FAN [5].

References

[1] H. Fujiwara, *Logic Testing and Design for Testability*, The MIT Press, 1985.

[2] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits," *Proc. IEEE Int. Symp. Circuits and Systems: Special Session on ATPG and Fault Simulation*, June 1985.

[3] J. P. Roth, "Diagnosis of automata failures: A calculus and a method," *IBM J. Res. Develop.*, vol. 10, pp. 278-291, July 1966.

[4] P. Goel, "An implicit enumeration algorithm to generate tests for combinational logic circuits," *IEEE Trans. Comput.*, vol. C-30, pp. 215-222, Mar. 1981.

[5] H. Fujiwara and T. Shimono, "On the acceleration of test generation algorithms," *IEEE Trans. Comput.*, vol. C-32, pp. 1137-1144, Dec. 1983.

[6] H. Fujiwara, "FAN: A fanout-oriented test pattern generation algorithm," *Proc. IEEE Int. Symp. Circuits and Systems*, pp. 671-674, June 1985.

[7] T. Kirkland and M. R. Mercer, "A topological search algorithm for ATPG," *Proc. 24th ACM/IEEE Design Automation Conf.*, pp. 502-508, June 1987.

[8] M. Schulz, E. Trischler and T. M. Sarfert, "SOCRATES: A highly efficient automatic test pattern generation system," *IEEE Trans. Computer-Aided Design*, vol. 7, pp. 126-137, Jan. 1988.

[9] M. Schulz and E. Auth, "Improved deterministic test pattern generation with applications to redundancy identification," *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 811-816, July 1989.

[10] T.Larrabee, "Efficient generation of test patterns using Boolean difference," *Proc. IEEE Int. Test Conf.*, pp. 795-801, Aug. 1989.

[11] J. Rajski and H. Cox, "A method to calculate necessary assignments in algorithmic test pattern generation," *Proc. IEEE Int. Test Conf.*, pp. 25-34, Sept. 1990.

[12] J. Giraldi and M. L. Bushnell, "EST: The new frontier in automatic test-pattern generation," *Proc. 27th ACM/IEEE Design Automation Conf.*, pp. 667-672, June 1990.