III. Conclusions

Accuracy of switch level modeling for VLSI testing has been examined. SLA generally predicts the correct logic output, although it does not predict any degradation in noise margins. In some cases, it will predict a definite value even though the actual voltage stays outside the noise margin.

When the substrate is connected to the source, there are some cases in which results from the SLA do not match with those from the circuit level analysis. For faults like gate to source, gate to drain, and drain to source shorts, the switch level model is adequate for obtaining the necessary test vectors. For shorts between a logical input and a nonlogical node of another column, the switch level model does not correctly predict the behavior under all vectors. In these cases an analysis at a level below the switch level is essential. Also, when a short occurs between two nodes of two different PCC's, the switch level model may not be accurate enough to generate the necessary test vectors. Under some specific conditions, the ARC phenomenon must be taken into account.

SLA can be used very effectively to characterize faults. However, in some specific cases, the circuit level behavior must be taken into account for accuracy.

Acknowledgment

The authors acknowledge the discussions with Dr. D. Ellsworth and E. Marchand of NCR, Fort Collins. The authors are thankful to the reviewers for their numerous suggestions and to Prof. C. W. Wilmsen of Colorado State University for his comments.

References

[1] J. Galiay, Y. Crouzet, and M. Vergnianlt, "Physical versus logical fault models in MOS LSI circuits, impact on their testability," in *Proc. Int. Symp. Fault-Tolerant Computing*, pp. 195–202, June 1979.

[2] Y. K. Malaiya, A. P. Jayasumana and R. Rajsuman, "A detailed study of bridging faults," in *Proc. IEEE Int. Conf. Computer Design*, pp. 78–82, 1986.

[3] S. A. Al-Arian and D. P. Agrawal, "CMOS fault testing," in *Proc. IEEE Test Conf.*, pp. 218–223, 1984.

[4] J. P. Hayes, "A fault simulation methodology for VLSI," in *Proc. 19th Design Automation Conf.*, pp. 393–399, 1982.

[5] P. Agrawal, "Test generation at switch level," in *Proc. IEEE Int. Conf. Computer-Aided Design*, pp. 128–130, 1984.

[6] K. W. Chiang and Z. G. Vranesic, "Test generation for MOS complex gate networks," in *Proc. Int. Symp. Fault Tolerant Computing*, pp. 149–157, 1982.

[7] H. C. Shin and J. A. Abraham, "Transistor level test generation for physical failures in CMOS circuits," in *Proc. 23rd Design Automation Conf.*, pp. 243–249, 1986.

[8] Y. K. Malaiya and S. Y. H. Su, "A new fault model and testing technique for CMOS devices," in *Proc. Int. Test Conf.*, pp. 25–34, 1982.

[9] T. W. Houston, "Considerations for the design of an SRAM with SOI technology," *IEEE Circuit Devices Mag.*, vol. 3, pp. 8–10, Nov. 1987.

[10] J. G. Fossum, R. Sundaresan, and M. Matloubian, "Anomalous subthreshold current-voltage characteristics of n-channel SOI MOSFET's," *IEEE Electron Device Lett.*, vol. EDL-8, pp. 544–546, Nov. 1987.

[11] J. R. Davis, A. E. Glaccum, K. Reeson, and L. F. Hemment, "Improved subthreshold characteristics of n-channel SOI transistors," *IEEE Electron Device Lett.*, vol. EDL-7, pp. 570–572, Oct. 1986.

[12] N. Sasaki, "Charge pumping in SOS MOS transistors," *IEEE Trans. Electron Devices*, vol. ED-28, pp. 48–52, Jan. 1981.

[13] R. Rajsuman, Y. K. Malaiya, and A. P. Jayasumana, "On accuracy of switch level modeling for bridging faults in complex gates," in *Proc. 23rd Design Automation Conf.*, pp. 244–250, June 1987.

[14] R. Rajsuman, "Analysis and modeling of major failure modes in MOS VLSI," Ph.D. dissertation, Dep. Elect. Eng., Colorado State Univ., 1988.

Improved Deterministic Test Pattern Generation with Applications to Redundancy Identification

MICHAEL H. SCHULZ AND ELISABETH AUTH

Abstract—**Based upon the sophisticated strategies used in the automatic test pattern generation system SOCRATES, this paper presents several new concepts and techniques aiming at a further improvement and acceleration of the deterministic test pattern generation and redundancy identification process. In particular, we will describe an improved implication procedure and an improved unique sensitization procedure. While the improved implication procedure takes advantage of the dynamic application of a learning procedure, the improved unique sensitization procedure profits from a dynamic and careful consideration of the existing situation of value assignments in the circuit. As a result of the application of the proposed techniques, SOCRATES is capable of both successfully generating a test pattern for all testable faults in a set of combinational benchmark circuits, and of identifying all redundant faults with less than 10 backtrackings.**

I. Introduction

Recent advances in VLSI technology and the increasing economic impact of testing operations on the overall chip costs have led to the meanwhile widespread use of the various scan-design techniques [1]. Since those techniques offer the important advantage of reducing the problem of testing complex sequential circuits to that of testing combinational circuits, efficient methods for fault simulation and automatic test pattern generation (ATG) in combinational circuits have gained great practical significance. Motivated by this fact, a considerable number of ATG approaches for combinational circuits, as e.g., [2]–[8], has been proposed.

Moreover, in view of the increasing quality requirements, it has frequently become a necessity to identify all redundant faults in a given circuit, in order to guarantee the completeness of the generated test set with respect to all testable faults. As is well known, redundancy identification (RI) can be performed by both methods specifically developed for this purpose, as e.g., [9], as well as deterministic ATG algorithms, which are complete in the sense that, given enough time, they will generate a test pattern for all testable, i.e., non-redundant, faults.

Among the ATG approaches cited above, the most significant progress has been achieved by the ATG system SOCRATES [7], [8], which has been presented very recently. SOCRATES' efficiency, even in a workstation environment, results from the application of a very fast fault simulation algorithm [10] and a sophisticated deterministic ATG algorithm. The latter one benefits from several distinct techniques which optimize the pruning of the search space [11], that has to be investigated during the deterministic ATG process. As a consequence, conflicting value assignments are recognized much earlier and the number of occurring backtrackings is drastically reduced. In particular, the deterministic ATG algorithm of SOCRATES incorporates an improved implication procedure, an improved unique sensitization procedure, and an improved multiple backtrack procedure, which all have been developed by extending the concepts of the FAN algorithm [4].

A specific feature of the deterministic ATG algorithm implemented in SOCRATES consists in its capability of identifying most

Manuscript received May 25, 1988; revised November 24, 1988 and January 11, 1989. The review of this paper was arranged by Associate Editor V. K. Agarwal.

The authors are with the Institute of Computer-Aided Design, Department of Electrical Engineering, Technical University of Munich, D-8000 Munich 2, Germany.

IEEE Log Number 8927529.

redundant faults with negligible computational effort [7], [8]. However, since both ATG as well as RI belong to the class of NP-complete problems [12], [13], there still exist faults, for which even SOCRATES is not able to decide within reasonable amounts of CPU-time, whether they are redundant or not. In order to overcome this deficiency, this paper presents several new concepts, which further improve the deterministic ATG approach described in [7], [8] and which specifically aim at an acceleration of the ATG and RI process for those faults, for which it is difficult to generate a test pattern or which are hard to be proved redundant [14]. In particular, in order to increase the power of the improved implication procedure, we propose to apply the learning procedure [7], [8] dynamically during the deterministic ATG process, instead of restricting it to the preprocessing phase. Furthermore, the advantageous impact of the improved unique sensitization procedure is greatly enhanced by a dynamic examination of the existing situation of value assignments in the circuit. The proposed techniques have been implemented in the ATG system SOCRATES, which is now able to successfully generate a test pattern for all non-redundant stuck-at faults in the well-known combinational benchmark circuits [15] and to identify all redundant stuck-at faults with less than 10 backtrackings. Thus aborted faults do not exist any more and, consequently, the generated test sets are guaranteed to be complete with respect to all testable stuck-at faults.

II. Deterministic Test Pattern Generation and Redundancy Identification

As is well known, the problem of deterministic ATG can be viewed as a *finite space search problem* [3], [7]. Since its properties fulfill the requirements for a *systematic search* and the application of the *split-and-prune method* [11], deterministic ATG algorithms usually build up a *decision tree* and apply a *backtracking search procedure*, in order to generate a test pattern for the given target fault [3], [4], [7].

According to [7], the decision tree can be described abstractly by a triangle which, for a non-redundant fault, is usually divided up into one *solution area* and into several *non-solution areas* (Fig. 1(a)). On the contrary, for a redundant fault, no test pattern exists and, consequently, the whole decision tree consists of a single non-solution area (Fig. 1(b)). The main shortcoming deterministic ATG algorithms suffer from is that, in general, they are not able to identify the entire non-solution areas, but only parts of them. Whenever a deterministic ATG algorithm gets into one of the *identified non-solution areas*, which are marked by dark shading in Fig. 1, it will recognize that no test pattern can be generated for the target fault by additional assignments of logic values to the signals of the circuit. In this case, it will immediately resort to backtracking, i.e., reject already made assignments and choose a different alternative which has not yet been examined. Contrary to the identified non-solution areas, the *unidentified non-solution areas*, indicated by light shading in Fig. 1, entail the wastage of large amounts of CPU-time, since an enormous number of backtrackings may be required, in order to leave them. As a conclusion, it has been pointed out in [7], [8] that the minimization of the unidentified non-solution areas represents the most important global goal at which improvements of deterministic ATG algorithms have to aim. Moreover, the minimization of the unidentified non-solution areas also supports and accelerates the RI process, since significantly fewer backtrackings will be necessary, in order to verify that the whole decision tree consists of a single non-solution area.

In order to achieve this desired minimization of the unidentified non-solution areas, the pruning of the search space has to be optimized [11]. For this reason, modern ATG approaches, like [4] and [7], apply implication and unique sensitization procedures in each step of the algorithm. The basic task of these two procedures consists in the immediate assignment of all those logic values to the corresponding signals, which are:

1) uniquely determined by other existing signal values, or
2) indispensable for propagating the effects of the injected target fault to at least one of the circuit outputs.

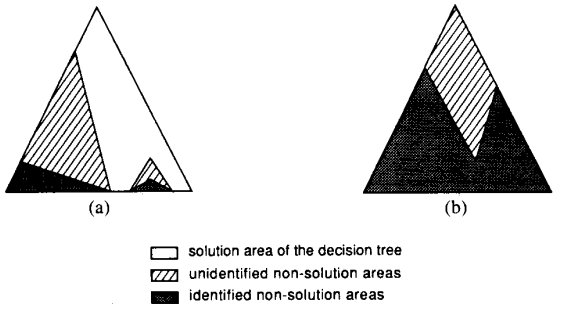

Fig. 1. Problem representation of deterministic ATG and RI. (a) Decision tree for a non-redundant fault. (b) Decision tree for a redundant fault.

Unfortunately, deterministic ATG algorithms are usually unable to identify *all* logic values fulfilling the conditions 1) or 2). Nevertheless, the instant assignment of the recognized ones essentially contributes to the desired reduction in search space and leads to an earlier detection of conflicts as well as to a decrease in the number of backtrackings. In order to enhance this advantageous impact, we have developed improved implication and unique sensitization procedures, which are capable of identifying a significantly increased number of logic values specified by the conditions 1) and 2).

III. The Improved Implication Procedure

3.1. Static Learning

While the implication procedure used in the FAN algorithm [4] is restricted to *local* implications, a learning procedure, which is carried out during a preprocessing phase, enables the improved implication procedure of SOCRATES to additionally perform *global* implications over reconvergent fanout [7], [8]. The basic strategy of this learning procedure is to:

1) assign a logic value to a certain signal of the circuit,
2) perform all implications from that assignment,
3) learn from the results of those implications.

These three steps are applied to *all* signals of the circuit for both logic values 0 and 1. Fig. 2 illustrates steps 1) and 2).

Thereby, let us assume that the current learning step has been initialized by the assignment $a = 1$. Subsequently, in step 2), the logic value 1 is assigned to the signals d, e, and f by the implication procedure. Step 3) of the learning procedure is based upon the well-known *logical identity* called the *law of contraposition* [16]:

$$(P \Rightarrow Q) \Leftrightarrow (\neg Q \Rightarrow \neg P). \tag{1}$$

Replacing the assertion variables P and Q in (1) by the assertions "the logic value of signal a is 1" and "the logic value of signal f is 1", respectively, we obtain

$$(a = 1) \Rightarrow (f = 1) \Leftrightarrow \neg(f = 1) \Rightarrow \neg(a = 1). \tag{2}$$

Executing the negations in the right side of (2), we finally get

$$(a = 1) \Rightarrow (f = 1) \Leftrightarrow (f = 0) \Rightarrow (a = 0). \tag{3}$$

As a result, the global implication

$$(f = 0) \Rightarrow (a = 0) \tag{4}$$

is learned in step 3) and, consequently, can be executed during the deterministic ATG process, if it is guaranteed that the injected target fault does not affect the part of the circuit shown in Fig. 2.

3.2. Dynamic Learning

For difficult faults in terms of both ATG as well as RI, we propose to apply the learning procedure *dynamically*, instead of executing it only once during the preprocessing phase [14]. As we will elaborate on, this leads to the identification of considerably more global implications and signal values, which are uniquely determined.

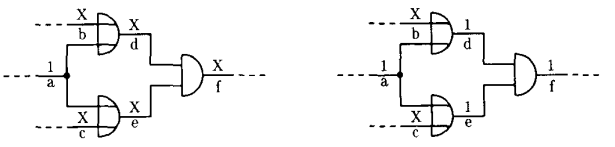

Fig. 2. Learning from $a = 1$.

Similar to the learning procedure employed in the preprocessing phase, the dynamic learning procedure basically executes steps 1)–3) *for all* those *signals* which still have the *unspecified logic value X* and which *cannot be reached* by the effects of the injected target fault. However, in order to completely exploit the results of the implications performed in step 2), we have developed four major concepts to be applied in step 3) of the dynamic learning procedure. For their explanation, we will assume that the learning step under consideration has been initialized at signal a in each of the illustrative examples presented in Figs. 3–5. Moreover, signal values, that have already been assigned at a preceding stage of the deterministic ATG process, will be shown within a box, as e.g., 1.

- *Concept 1 (Dynamic Learning of Global Implications):*

If both assignments $a = 0$ and $a = 1$ do not result in a conflict during the execution of the implication procedure in step 2), a learning criterion (for more details, see [7] or [8]) is applied, in order to determine those additional global implications that are worthwhile learning. Fig. 3(a) shows a part of a circuit, in which the learning procedure performed during the preprocessing phase is not able to learn any global implication. However, if the logic value 1 has been assigned to the signals b and c at an earlier stage of the deterministic ATG process, the assignment $a = 1$ will obviously imply the logic value 1 at the signals e, f, and h. Thus the global implication

$$(h = 0) \Rightarrow (a = 0) \tag{5}$$

is learned with the aid of the learning criterion and, consequently, can be taken advantage of later on, if the logic value 0 should be assigned to signal h.

- *Concept 2 (Identification of Uniquely Determined Signal Values):*

In addition, both assignments $a = 0$ and $a = 1$ lead to $i = 0$, if the injected target fault does not affect the part of the circuit shown in Fig. 3. On the basis of the *logical inference* called the *constructive dilemma* [16]:

$$[(a = 0) \Rightarrow (i = 0)] \wedge [(a = 1) \Rightarrow (i = 0)] \Rightarrow (i = 0) \tag{6}$$

we can immediately assign the logic value 0 to signal i, regardless of the value which signal a is going to assume during the further ATG process.

- *Concept 3 (Identification of Uniquely Determined Signal Values):*

Fig. 4 illustrates a different situation which also allows the identification of uniquely determined signal values. If the current learning step is initialized by the assignment $a = 0$, the implication procedure assigns the logic value 0 to the signals d and e (Fig. 4(a)). Obviously, this leads to a conflict at the OR gate with the output signal f, since the logic value 1 at signal f requires at least one of the gate inputs to have the logic value 1 as well. With the aid of another logical inference ("*modus tollens*", [16]):

$$(f = 1) \wedge [(a = 0) \Rightarrow (f = 0)] \Rightarrow (a = 1) \tag{7}$$

we can readily deduce that the logic value 1 must be assigned to signal a, in order to justify the logic value 1 at signal f (Fig. 4(b)).

- *Concept 4 (Recognition of the Necessity of Backtracking):*

In Fig. 5, the assignment $a = 0$ implies the logic value 0 at the signals f and g and, consequently, causes a conflict at the OR gate with the output signal k. Likewise, the assignment $a = 1$ produces a conflict at the AND gate with the output signal m. Since both assignments $a = 0$ and $a = 1$ result in conflicts, we can conclude

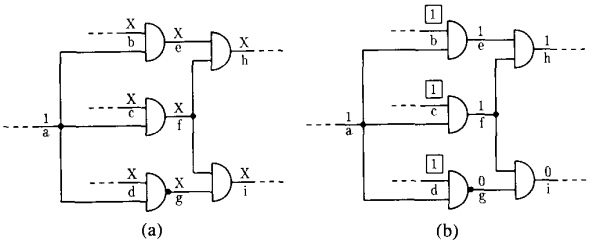

Fig. 3. Advantageous impact of the dynamic learning procedure; learning of additional global implications (concept 1) and identification of uniquely determined signal values (concept 2). (a) Preprocessing phase. (b) Dynamic learning.

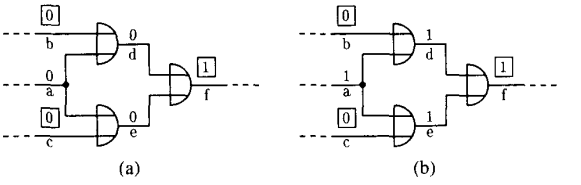

Fig. 4. Identification of uniquely determined signal values (concept 3). (a) Learning from $a = 0$. (b) Assignment of uniquely determined values.

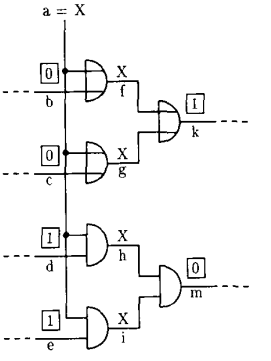

Fig. 5. Recognition of the necessity of backtracking (concept 4).

that the deterministic ATG algorithm has inadvertently entered a non-solution area of the decision tree (Fig. 1), which has not been identified up to now. Consequently, in order to leave this non-solution area as soon as possible and to avoid fruitless computations, we immediately have to perform backtracking.

Note that only concept 1 results in the explicit learning of global implications, while the concepts 2, 3, and 4 lead to immediately assigning uniquely determined logic values to distinct signals or to recognizing the necessity of backtracking. Moreover, all operations performed by the dynamic learning procedure are related to signal values that have already been assigned at an earlier stage of the deterministic ATG process. Consequently, both the learned global implications as well as the uniquely determined logic values have to be recorded in the decision tree and must be rejected, if the respective node of the decision tree should be deleted by backtracking.

To summarize, the identification of additional global implications, the immediate assignment of uniquely determined logic values to the corresponding signals, and the earlier recognition of the necessity of backtracking significantly optimize the pruning of the search space. As a consequence, the unidentified non-solution areas of the decision tree are successfully minimized, leading to a substantial decrease in the number of backtrackings to be executed. Finally, the method of learning dynamically, in a certain way, appears to be similar to the concept of dynamic testability measures [17].

IV. Advanced Dynamic Unique Sensitization Techniques

Based upon [4], an improved unique sensitization procedure has been presented in [7], [8], in order to identify and execute those value assignments which are indispensable for propagating the effects of the injected target fault to at least one of the circuit outputs. Its global goal is to improve the handling of those situations in which:

- the D-frontier consists of a *single* signal,
- *all paths* from the D-frontier to the primary outputs of the circuit pass through *distinct* gates.

Similar to another ATG approach proposed recently [18], the improved unique sensitization procedure makes use of the notion of dominance between the signals of a combinational circuit [19], in order to identify the situations characterized above. The determination of the dominance relationships is performed by an analysis of the circuit structure during the preprocessing phase [7]. Taking the concept of dominance as the basis and relying mainly on the results of that analysis of the circuit structure, the improved unique sensitization procedure comprises two major instructions [7], [8]. Although instruction 2 represents a considerable improvement over the FAN algorithm [4], the power of the unique sensitization procedure can be further enhanced, if the number of logic values, that are identified to be indispensable for propagating the effects of the target fault to at least one of the primary outputs, is increased. This can be achieved by a *dynamic* examination of the existing situation of value assignments in the circuit [14].

For this purpose, we first introduce the definitions of the potential propagation paths and the dynamic dominance relationships.

Definition 1: A directed path of signals $P = (x_1, x_2, \cdots, x_n)$ with $x_1 \neq x_n$ is called a *potential propagation path*, if signal x_1 has the logic value D or $\overline{D}$ and all other signals $x_2, \cdots, x_n$ on P have the unspecified logic value X.

Definition 2: A signal y is said to be a *dynamic dominator* of signal x, i.e., $y \in dyn\_dom(x)$, if all *potential propagation paths* from x to the primary outputs of the circuit pass through signal y.

In Fig. 6(a), for example, $(a, e, h, \cdots)$ and $(a, f, h, \cdots)$ represent potential propagation paths, while $(a, d, \cdots)$ does not. Consequently, although signal a has no structural dominator, signal h represents a dynamic dominator of signal a, i.e., $h \in dyn\_dom(a)$. On the basis of the definitions 1 and 2, the *dynamic* instructions 1 and 2 of the improved unique sensitization procedure can be formulated.

- *Dynamic Instruction 1:*

Let the D-frontier consist of the signals $x_1, \cdots, x_n$ and let $Y = \{y_1, \cdots, y_m\}$ denote the set of those dynamic dominators which are *common to all* signals $x_1, \cdots, x_n$, i.e.:

$$Y = dyn\_dom(x_1) \cap \cdots \cap dyn\_dom(x_n). \tag{8}$$

Furthermore, let the elements of Y be the respective output signals of the gates combined in the set $G = \{g_1, \cdots, g_m\}$. Then, for all gates $g \in G$, the *non-controlling* value is assigned to all those inputs of g that *cannot be reached* from any signal in the D-frontier on a *potential propagation path.*

- *Dynamic Instruction 2:*

Let y be an element of Y (defined by (8)) or the only signal the D-frontier consists of. Furthermore, let $g_1, \cdots, g_n$ denote those gates which are fed by signal y and which are located on at least one potential propagation path between the signals in the D-frontier and the primary outputs of the circuit. If *all* gates $g_1, \cdots, g_n$ require the *same non-controlling value* $v \in \{0, 1\}$, then v is assigned to all those signals $z \neq y$ which branch out to *all* gates $g_1, \cdots, g_n$, too.

Fig. 6(b) and (c) illustrate our motivation for developing the dynamic instructions 1 and 2 and demonstrate their practical application. First of all, it should be explicitly mentioned that, by exploiting the dynamic instructions 1 and 2, the improved unique sensitization procedure is no longer restricted to those situations in which there is only one signal in the D-frontier, but can also be successfully applied to more general situations. However, for the sake of simplicity, in Fig. 6 we again assume that the D-frontier consists of signal a only.

Since signal a has no structural dominator, the structure-based instruction 1 [7], [8] is not able to perform any value assignment in the shown circuit. However, since the logic value 1 has already been assigned to the signals b and d at a preceding stage of the deterministic ATG process, no potential propagation path exists through signal d. As a consequence, signal h represents a dynamic dominator of signal a, implying that the effects of the target fault must be propagated over signal h, in order to reach one of the circuit outputs. Therefore, the logic value 1 is immediately assigned to signal g by the dynamic instruction 1, since signal g is not reachable from signal a on a potential propagation path (Fig. 6(b)). In addition, due to the facts that the OR gate with the output signal d is not fed by signal c and that it requires a different non-controlling value than the two AND gates with the output signals e and f, the conditions for the application of the structure-based instruction 2 [7], [8] are not fulfilled. However, as we have already established, no potential propagation path exists through signal d. Consequently, a path has to be sensitized through at least one of the signals e or f. Since both AND gates with the output signals e and f require the same non-controlling value 1 and since both of them are fed by signal c, this can only be achieved by the assignment $c = 1$, which is performed by the dynamic instruction 2 (Fig. 6(c)).

Finally, it is worth noting that, in this particular situation, the effects of the target fault have been propagated from signal a to signal h with the aid of the dynamic instructions 1 and 2 (Fig. 6(d)), whereas the structure-based instructions 1 and 2 [7], [8] would not be able to perform any of the two value assignments $g = 1$ and $c = 1$, which are both indispensable for successfully generating a test pattern for the target fault. This fact substantiates that the dynamic instructions 1 and 2 considerably enhance the advantageous impact of the improved unique sensitization procedure and, thus essentially contribute to the desired minimization of the unidentified non-solution areas in the decision tree. As the dynamic application of the learning procedure, they are of particular use for faults causing difficulties in terms of both ATG as well as RI. Moreover, the dynamic instructions 1 and 2 offer the welcome advantage that they can be implemented with an almost negligible computational overhead in terms of both extra code and additional memory resources.

V. Experimental Benchmark Results

After implementing the proposed techniques in the deterministic ATG algorithm of SOCRATES, we have applied SOCRATES to the ten well-known combinational benchmark circuits [15]. Thereby, the overall ATG process has been performed in four phases.

Phase RTG: random test pattern generation;
Phase STAT: deterministic ATG, using only the global implications learned in the preprocessing phase and applying the structure-based unique sensitization techniques;
Phase DYN_1: deterministic ATG, using only the global implications learned in the preprocessing phase, but applying the dynamic instructions 1 and 2 in the improved unique sensitization procedure;
Phase DYN_2: deterministic ATG, invoking the learning procedure dynamically each time after creating a new node in the decision tree and applying the dynamic instructions 1 and 2 in the improved unique sensitization procedure.

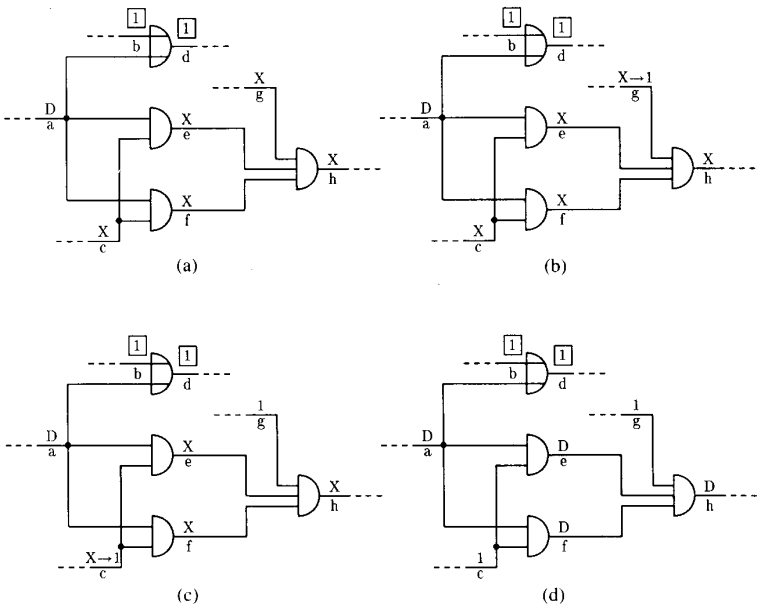

Fig. 6. Application of the dynamic instructions 1 and 2 of the improved unique sensitization procedure. (a) Initial situation. (b) Dynamic instruction 1: assignment $g = 1$. (c) Dynamic instruction 2: assignment $c = 1$. (d) Final situation: fault effects propagated from signal a to signal h.

In phase RTG, the random patterns have been generated in portions of 32 patterns and subjected to a highly efficient fault simulator, which is capable of applying parallel processing of patterns at all stages of the calculation procedure [10]. Phase RTG has been stopped, if 64 succeeding random patterns did not detect any additional uncovered fault. Subsequently, the backtracking limit has been set to 10 in each of the three deterministic ATG phases STAT, DYN_1, and DYN_2. Naturally, in each phase only those faults have been taken into consideration that have not been detected and that have not been proved to be redundant in a preceding phase. In addition, in order to achieve a compaction of the generated test set, a reverse order fault simulation has been performed at the final stage of the entire ATG process. Table I summarizes the results, achieved on an APOLLO DN 3000 workstation.

The results shown in Table I demonstrate the efficiency of SOCRATES very impressively. Even for the largest circuit c7552, SOCRATES generates a test set which is guaranteed to be complete with respect to all non-redundant stuck-at faults in approximately 4 CPU-min. Moreover, not less remarkably, all redundant stuck-at faults in the ten benchmark circuits are identified with less than 10 backtrackings. Consequently, no aborted faults, i.e., faults for which the deterministic ATG algorithm is unable to decide whether they are redundant or not, remain. Finally, the numbers of occurred backtrackings appear to be negligible in comparison to most other ATG approaches and a considerable number of redundant faults is identified without any backtracking (e.g., circuit c3540).

A more detailed analysis reveals that, for eight of the ten benchmark circuits, the ATG and RI process is already completed after the first two phases. At this time, only two faults of circuit c432 and 55 faults of circuit c7552 are not detected and not proved to be redundant. Subsequently, in phase DYN_1, 54 of the 55 previously aborted faults of circuit c7552 turn out to be redundant, while a test pattern is successfully generated for the remaining one. For all 55 faults, no backtrackings occur in phase DYN_1, and the total CPU-time required by the deterministic ATG algorithm amounts to 13.9 s on an APOLLO DN 3000 workstation. Finally, both faults of circuit c432, that have been aborted in the phases STAT and DYN_1, are proved to be redundant without any backtracking in phase DYN_2, using a total CPU-time of 0.8 s. Thereby, 172 global implications are learned dynamically with the aid of concept 1, and 72 uniquely determined signal values are assigned to the corresponding signals by employing the concepts 2 and 3. Furthermore, for both faults, the application of concept 4 results in establishing their redundancy already during the creation of the root node of the respective decision tree. Thus no backtracking is necessary for proving those two faults to be redundant.

TABLE I
RESULTS OF ATG AND RI BY SOCRATES (APOLLO DN 3000)

Circuit Name	Fault Coverage [%]	Test Patterns	Aborted Faults	Redundant Faults	Backtrackings	CPU-Time [sec.] ATG	Fault Sim.	Total
c432	99.24	53	0	4	40	2.9	1.4	4.3
c499	98.94	54	0	8	0	2.6	2.3	4.9
c880	100.00	70	0	0	0	1.4	3.8	5.2
c1355	99.49	87	0	8	5	6.9	7.0	13.9
c1908	99.52	123	0	9	0	16.5	17.3	33.8
c2670	95.49	123	0	117	8	35.7	21.8	57.5
c3540	96.00	184	0	137	0	22.1	34.5	56.6
c5315	98.90	149	0	59	6	8.3	23.0	31.3
c6288	99.56	38	0	34	0	1.8	85.2	87.0
c7552	98.26	236	0	131	615	172.0	75.6	247.6

In order to illustrate the advantageous impact of the techniques presented in this paper, we have performed the RI process for the two originally aborted faults of circuit c432 by three different versions of the deterministic ATG algorithm. While the versions DYN_2 and STAT are identical with those applied in the corresponding phases of the ATG process described above, FAN refers to an implementation which uses the conventional implication and unique sensitization procedures of the FAN algorithm [4]. Table II compares the numbers of backtrackings and the CPU-times that have been required by DYN_2, STAT, and FAN, in order to prove those two faults to be redundant. By applying the learning proce-

TABLE II
RI FOR THE TWO ORIGINALLY ABORTED FAULTS OF CIRCUIT c432; ADVANTAGEOUS IMPACT OF THE DYNAMIC APPLICATION OF THE LEARNING PROCEDURE AND THE DYNAMIC INSTRUCTIONS 1 AND 2 OF THE IMPROVED UNIQUE SENSITIZATION PROCEDURE (APOLLO DN 3000)

Version	Number of Backtrackings	CPU-Time [sec.] for RI
DYN_2	0	0.8
STAT	16567	145.8
FAN	76911	785.6

TABLE III
RESULTS OF ATG AND RI BY SOCRATES WITHOUT APPLYING RTG (APOLLO DN 3000)

Circuit Name	Fault Coverage [%]	Test Patterns	Aborted Faults	Redundant Faults	Backtrackings	CPU-Time [sec.] ATG	Fault Sim.	Total
c432	99.24	61	0	4	50	6.4	3.1	9.5
c499	98.94	60	0	8	35	24.4	4.7	29.1
c880	100.00	59	0	0	0	4.0	6.1	10.1
c1355	99.49	94	0	8	62	30.0	14.1	44.1
c1908	99.52	127	0	9	21	53.4	27.4	80.8
c2670	95.49	123	0	117	8	48.0	31.6	79.6
c3540	96.00	191	0	137	31	85.8	72.8	158.6
c5315	98.90	134	0	59	33	39.4	63.0	102.4
c6288	99.56	44	0	34	21	26.0	112.7	138.7
c7552	98.26	224	0	131	696	276.3	142.1	418.4

dure dynamically and taking advantage of the dynamic instructions 1 and 2 of the improved unique sensitization procedure, version DYN_2 accelerates the RI process by a factor of 182 in comparison to version STAT and even by a factor of 982 in comparison to version FAN.

Furthermore, we randomly selected one of the 54 faults of circuit c7552 that have been identified to be redundant in phase DYN_1. For this particular fault, we tried to perform the RI process by version STAT, which does not exploit the dynamic instuctions 1 and 2 of the unique sensitization procedure. While no backtrackings and only 0.32 s of CPU-time have been required in phase DYN_1 for proving this fault to be redundant, version STAT capitulates without success after executing 1 000 000 backtrackings and wasting almost 9 h of CPU-time.

Finally, in a futher experiment, we have performed ATG for the ten benchmark circuits using only the three deterministic ATG phases STAT, DYN_1, and DYN_2, but skipping phase RTG. The results summarized in Table III substantiate that suppressing phase RTG does not have any significant impact on the overall efficiency of SOCRATES. Although the CPU-times are slightly higher than those reported in Table I, SOCRATES still succeeds in generating test patterns for all testable stuck-at faults and in identifying all redundant stuck-at faults with remarkably small computational effort.

However, concerning this experiment of skipping phase RTG, it is interesting to note that, besides the circuits c432 and c7552, phase DYN_1 has also to be employed for one fault of circuit c3540 and for one fault of circuit c5315. For both faults, a test pattern is found in phase DYN_1, requiring 1.2 and 0.4 CPU-s, respectively.

VI. CONCLUSIONS

We have presented several new concepts and techniques which significantly improve and accelerate the deterministic ATG and RI process for those faults, for which it is difficult to generate a test pattern or which are hard to be proved redundant. In particular, an improved implication procedure, which takes advantage of the dynamic execution of a learning procedure, and an improved unique sensitization procedure, profiting from the dynamic and careful investigation of the existing situation of value assignments in the circuit, have been described in detail. As a result of the implementation of the proposed techniques, SOCRATES is now capable of successfully generating a test pattern for all testable faults in the well-known benchmark circuits as well as of identifying all redundant faults with less than 10 backtrackings. Moreover, a number of illustrative comparisons has demonstrated that the methods presented in this paper can reduce the CPU-times required for ATG and RI by several orders of magnitude.

ACKNOWLEDGMENT

The authors are very grateful to Prof. K. Antreich of the Technical University of Munich for his valuable suggestions and his helpful advice and discussions, and to Dr. A. Sauer, E. Trischler, and T. Sarfert of Siemens, Dept. E STE 4, for their continuous support and encouragement.

REFERENCES

[1] E. J. McCluskey, *Logic Design Principles*. Englewood Cliffs, NJ: Prentice-Hall, 1986.
[2] J. P. Roth, *Computer Logic, Testing, and Verification*. Potomac, MD: Computer Science, 1980.
[3] P. Goel, "An implicit enumeration algorithm to generate tests for combinational logic circuits," *IEEE Trans. Comput.*, vol. C-30, pp. 215-222, Mar. 1981.
[4] H. Fujiwara and T. Shimono, "On the acceleration of test generation algorithms," *IEEE Trans. Comput.*, vol. C-32, pp. 1137-1144, Dec. 1983.
[5] M. Abramovici, J. J. Kulikowski, P. R. Menon, and D. T. Miller, "SMART and FAST: Test generation for VLSI scan-design circuits," *IEEE Design Test*, pp. 43-54, Aug. 1986.
[6] E. Trischler, "ATWIG, An automatic test pattern generator with inherent guidance," in *Proc. 1984 Int. Test. Conf.*, pp. 80-87, Oct. 1984.
[7] M. H. Schulz, E. Trischler, and T. M. Sarfert, "SOCRATES: A highly efficient automatic test pattern generation system," in *Proc. 1987 Int. Test Conf.*, pp. 1016-1026, Sept. 1987.
[8] ——, "SOCRATES: A highly efficient automatic test pattern generation system," *IEEE Trans. Computer-Aided Design*, vol. 7, pp. 126-137, Jan. 1988.
[9] D. Brand, "Redundancy and don't cares in logic synthesis," *IEEE Trans. Comput.*, vol. C-32, pp. 947-952, Oct. 1983.
[10] K. J. Antreich and M. H. Schulz, "Accelerated fault simulation and fault grading in combinational circuits," *IEEE Trans. Computer-Aided Design*, vol. CAD-6, pp. 704-712, Sept. 1987.
[11] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Reading, MA: Addison-Wesley, 1984.
[12] O. H. Ibarra and S. K. Sahni, "Polynomially complete fault detection problems," *IEEE Trans. Comput.*, vol. C-25, pp. 242-249, Mar. 1975.
[13] H. Fujiwara and S. Toida, "The complexity of fault detection problems for combinational logic circuits," *IEEE Trans. Comput.*, vol. C-31, pp. 555-560, June 1982.
[14] M. H. Schulz and E. Auth, "Advanced automatic test pattern generation and redundancy identification techniques," in *18th Int. Symp. Fault-Tolerant Computing*, pp. 30-35, June 1988.
[15] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a target translator in Fortran," *Proc. IEEE Int. Symp. Circuits and Systems*; (*Special Session on ATPG and Fault Simulation*), June 1985.
[16] A. Church, *Introduction to Mathematical Logic*. Princeton, NJ: Princeton Univ., 1956.
[17] A. Ivanov and V. K. Agarwal, "Testability measures—What do they do for ATPG?," in *Proc. 1986 Int. Test Conf.*, pp. 129-138, Sept. 1986.
[18] T. Kirkland and M. R. Mercer, "A topological search algorithm for ATPG," in *Proc. 24th Design Automation Conf.*, pp. 502-508, June 1987.
[19] R. E. Tarjan, "Finding dominators in directed graphs," *SIAM J. Comput.*, pp. 62-89, 1974.