CONTEST: A Fast ATPG Tool for Very Large Combinational Circuits

Udo Mahlstedt Torsten Grüning Cengiz Özcan Wilfried Daehn*

Institut für Theoretische Elektrotechnik
Universität Hannover
Appelstr. 9A, D-3000 Hannover 1, Germany

Abstract

In this paper CONTEST (CONe-oriented TEST pattern generator), a new ATPG tool for very large combinational digital circuits is presented. CONTEST is based on four major ideas. Cone-oriented circuit partitioning reduces the circuit complexity and increases the number of dominators. The propagation graph is a dynamic data structure that keeps track of all paths from the fault location to a primary output. The new multiple backtrace procedure reduces contradictory node assignments by examination of fanout nodes and dynamic implications. The new pattern parallel fault dropping technique is based on Hamming distance variations of generated test patterns. Experimental results for the ISCAS'85 and ISCAS'89 benchmark circuits containing up to 40.000 nodes illustrate the superiority of the ATPG system. For these circuits a 100 percent fault coverage for all detectable stuck-at faults and a 100 percent redundancy identification is achieved.

1 Introduction

To simplify the process of test pattern generation, several techniques known as design for testability have been developed [1,2,3] which reduce the sequential to a combinational test problem. But even for combinational circuits the problem of ATPG is known to be NP-complete.

Most existing state of the art algorithms [4,5,6,7,8] adopt a branch-and-bound technique to implicitely but exhaustively examine all input combinations. Since the search space can be quite large concepts have been developed which reduce this search problem. Two basic strategies exist which make the test generation problem more tractable:
1. The number of nodes appearing principally in the decision tree is limited to a restricted set of nodes as for example the primary inputs in the PODEM-algorithm [5] or the head lines in the FAN-algorithm [6]. This reduces the size of the decision tree and therefore the number of backtracks.
2. The determination of as many uniquely defined values as possible with a tenable amount of time at each stage of the test pattern generation process. In such a way the search space is drastically decreased.

The algorithms proposed in [6,7,8,9] use the concept of dominators [10] to determine mandatory node assignments for fault propagation. By definition the dominators of a faulty node consist of the faulty node itself and all nodes which the fault effect must go through to reach a primary output (PO). Since typically there are many primary outputs to which the fault effect can be propagated the number of dominators is rather small and therefore only few nodes can be set uniquely. This observation leads to an efficient cone-oriented circuit partitioning method [9] which significantly speeds up automatic test pattern generation.

In the next chapter we will explain the underlying ideas which contribute to the performance of the deterministic test pattern generator used in the ATPG-system CONTEST. In the third chapter we will present a new fault dropping technique based on Hamming distance variations of generated test patterns. The fourth chapter gives a brief overview of CONTEST. After that some experimental results are given. The paper will finish with a short conclusion and some comments on further work.

* Since September 1990 Dr. Daehn is working at SICAN GmbH, Hannover, Germany

2 Deterministic Test Pattern Generation

In opposite to other algorithms which handle the whole, i.e. unpartitioned, circuit CONTEST benefits from a partitioning technique which cuts a circuit into several cones. Thereby the search space is splitted thus simplifying the ATPG process. A cone is the fraction of a combinational circuit which includes all signal lines affecting one output. The underlying idea is that the number of dominators within each cone increases compared to the unpartitioned circuit. Experimental results for the ISCAS'85 benchmark circuits show an average increase in the number of dominators in the partitioned case by a factor of 2.1 [9]. The number of dominators of a faulty node can be considered as a guideline for the number of mandatory node assignments to propagate the fault effect to a primary output. Since these assignments are not a matter of choice they cannot cause any backtracks thus speeding up ATPG [9].

Like other ATPG algorithms CONTEST adopts a branch-and-bound technique. To reduce the size of the decision tree and thus the possible number of backtracks optional node assignments are only allowed at decision lines. A decision line is a point of absolute convergence of one or several primary inputs which can be controlled to 0 as well as to 1. A decision line is either a free line [6], or a basis line [7] which can be set to both signal values. For a given stuck-at fault only a subset of all decision lines is required for optional node assignments. Since the decision lines can be controlled independently the decision tree may not contain any inconsistent node assignments.

In opposite to the FAN-algorithm CONTEST performs no optional node assignments at internal fanout nodes. If a contradictory requirement at a fanout node occurs CONTEST performs an examination of this node to ensure that the decision chosen by the multiple backtrace procedure is consistent. This examination is done by evaluating the effects of a logical 0 respectively 1 at the concerned node. Based upon logical inferences [11] also used by SOCRATES [8] in a different context CONTEST is able to determine additional uniquely defined values in the following situations:
1. The logical value L at node N_1 results in an inconsistency at node N_2. Thus node N_1 must be set to the opposite value of L.
2. The logical value L at node N_1 disables further fault propagation. It follows that node N_1 must be set to the opposite value of L to enable fault propagation.
3. Both logical values 0 and 1 at node N_1 produce the same value assignment L at node N_2. Thus node N_2 must be set to L irrespectively of the value at node N_1.

The new multiple backtrace procedure examining fanout nodes with contradictory requirements leads to an acceleration of test pattern generation and an easier redundancy identification.

Figure 1 shows the flowchart of CONTEST. First a cone not yet be processed is selected and structural informations of the cone are determined. Then for each basis line the input signals to justify a logical 0 respectively 1 are generated and stored. By this way the basis lines can be justified without any backtracks just like free lines. If for any of the basis lines no input signals can be generated, either for the value 0 or 1, this basis line cannot be used as a decision line.

Having chosen a target fault belonging to the considered cone the sensitization of the fault is performed. The next step is to build the propaga-

CH2924-9/90/0000/0222$01.00 © 1990 IEEE

tion graph. The propagation graph consists of all paths on which the fault effect can be propagated to a primary output with respect to existing value assignments. The propagation graph is an efficient dynamic data structure to control the fault propagation. It enables an easy determination of the dominators at each stage of the test pattern generation process. Unlike other algorithms the determination of dominators and associated mandatory node assignments in CONTEST is not restricted to special situations.

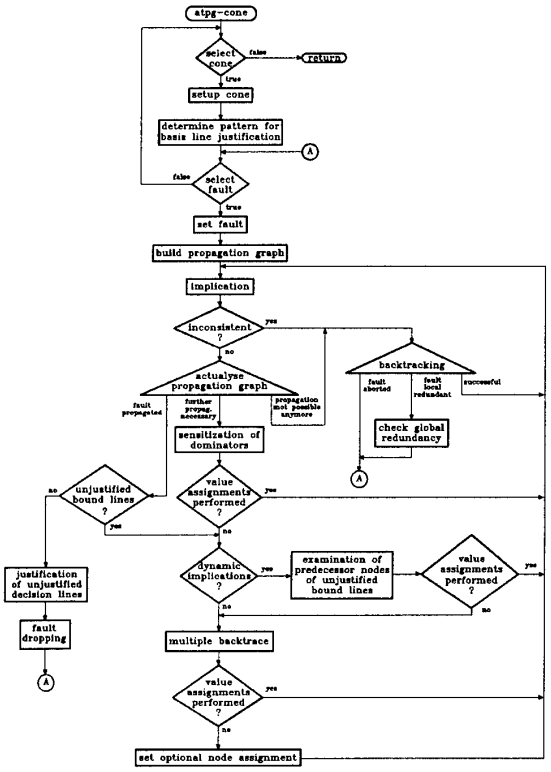

Fig 1: Flowchart of CONTEST

The implication procedure performs all forward and backward implications which are possible in the present situation. In opposite to most existing ATPG algorithms which use a five-valued logic CONTEST makes use of a nine-valued logic [12]. Together with the five signal values in the D-algorithm (s00, s11, sXX, s10 and s01) the four partly specified values s0X, s1X, sX0 and sX1 express more efficiently the state of signal nodes. For example the value s10 represents the logical state 1 in the good circuit and 0 in the faulty circuit. The nine-valued logic facilitates powerful implications especially at dominators.

If an inconsistency occurs during implication a backtrack has to be carried out. The last optional node assignment is removed until an untried possibility is found or the backtrack limit is reached. In the first case the backtracking has been successful and the algorithm continues with the implication procedure. In the second case the fault is aborted and the next target fault is selected. If no choice remains to perform a backtrack, i.e. the decision tree is empty, the target fault is known to be local redundant. Local redundant means that the fault cannot be detected in the processed cone but it might be possible that the fault can be observed at the PO of another cone. Since in most cases a local redundant fault is not detectable at all CONTEST tries to proof the global redundancy of the fault. By this way it is avoided that the fault at the considered node must be proved to be local redundant in each cone which it belongs to.

If no inconsistency occurs during implication the propagation graph is actualized. If the propagation graph disappears, that is no fault propagation is possible anymore, a backtrack must be performed. If the fault effect has not yet reached the PO the dominators are examined to identify mandatory node assignments to enable fault propagation. If value assignments could be performed at the dominators the algorithm continues with the implication procedure. Otherwise dynamic implications are determined or the multiple backtrace procedure is started. If the fault effect has already reached the PO and there are still unjustified bound lines the algorithm proceeds in the same manner.

The determination of dynamic implications is an optional feature which is used in a second run to generate tests for hard to detect faults or for redundancy identification. The determination of dynamic implications is based on the examination of predecessor nodes of unjustified boundlines. This is performed in the same manner as the examination of fanout nodes during backtracing.

If there are no more mandatory node assignments which can be performed the multiple backtrace procedure is started to choose a decision line for the next optional node assignment. If no node assignments, resulting from the examination of fanout nodes, have been carried out a new node is added to the decision tree. Otherwise the algorithms proceeds immediately with the implication procedure.

If the fault effect has reached a PO and no unjustified bound lines remain a test pattern is generated. After justifying the unjustified decision lines a fault dropping is performed and the next target fault is selected.

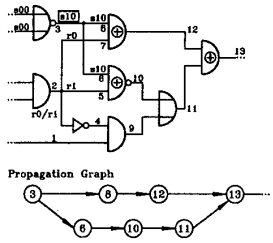

Fig. 2: Circuit graph after fault sensitization and implications

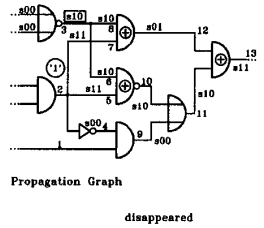

Fig. 3: Examination of node 2 for a logical '1'

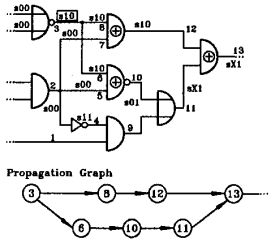

Fig. 4: Circuit graph after assigning a logical '0' to node 2 and the resulting implications

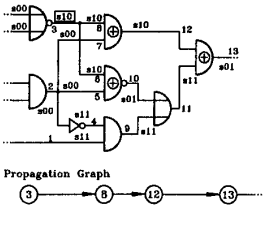

Fig. 5: Circuit graph after the mandatory assignments at node 13 and the resulting backward implications

Fig. 2-5 show an example illustrating the main new features of the deterministic test pattern generation. The given fault for this example is node 3 stuck-at 0, represented by 3/s10. After sensitizing the fault and performing implications we receive the situation shown in figure 2. The associated propagation graph is shown below.

The next step is the start of the multiple backtracing:

1. To propagate the fault through node 12 it is assumed that a 0 is requested at node 7 (represented by 7/r0).
2. To propagate the fault through node 10 it is assumed that a 1 is requested at node 5 (5/r1).

Further backtracing produces the contradictory requirement 2/r0,r1. Now node 2 has to be examined. The examination of node 2 for the logical value 1 (figure 3) proceeds as follows:

1) 2/s11 → 4/s00 → 9/s00
2) 2/s11 → 5/s11 → 10/s10 → 11/s10
3) 2/s11 → 7/s11 → 12/s01 → 13/s11

At this point the propagation graph disappears since node 13 takes on the value 1. It follows that node 2 must be set to 0 to enable fault propagation. This leads to the following signal values (figure 4):

1) 2/s00 → 4/s11
2) 2/s00 → 5/s00 → 10/s01 → 11/sX1
3) 2/s00 → 7/s00 → 12/s10 → 13/sX1

After actualizing the propagation graph the dominators are examined. Since only the value s01 at node 13 enables further fault propagation this node is uniquely defined. The value s01 at node 13 yields following implications:

1) 13/s01 → 11/s11 → 9/s11 → 1/s11

The resulting signal values for the given example are shown in figure 5. In this example CONTEST has been able to propagate the fault effect through node 13 without any optional node assignment and therefore without any backtracking. Other algorithms would fail to identify all mandatory node assignments in this situation.

3 Pattern Parallel Fault Dropping

To reduce the number of faults for which a deterministic test pattern generation (DTPG) has to be carried out most ATPG systems start with a random test pattern generation (RTPG) using a pattern parallel fault simulation [13]. Although CONTEST is also capable to do so there is no need for a RTPG before starting the DTPG since CONTEST uses a special fault dropping technique.

Usually ATPG systems perform a single pattern fault simulation with each generated test pattern. In contrast CONTEST adopts a pattern parallel fault simulation for fault dropping. Since only one bit of a machine word is necessary to simulate the generated test pattern the remaining bits are used to simulate random patterns at the same time. In this way the same effect as a preceded RTPG is achieved with no significant increase in time compared to the conventional DTPG with single pattern fault dropping. On the other hand the total amount of time needed for DTPG and RTPG can be reduced by the pattern parallel fault dropping.

In addition to the parallel simulation of random patterns during fault dropping CONTEST has a more powerful technique making use of the pattern parallel fault simulation. Instead of random patterns a special kind of patterns based on Hamming distance variations of the generated fully specified values at the primary inputs (PI) are used. The underlying idea of this technique is that the path sensitized by the generated test pattern remains almost unchanged so that faults not yet detected can be propagated along the same path. The sensitization of other faults thereby is achieved by randomly assigning values to the PIs which have been left unspecified during the deterministic test pattern generation process.

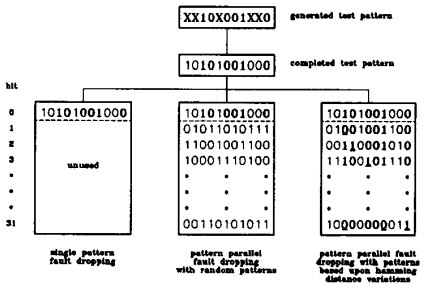

Fig. 6: Different fault dropping techniques

Figure 6 illustrates the different fault dropping techniques for a circuit with 11 primary inputs and a machine word length of 32 bits. After the DTPG we receive a test pattern with some PIs left unspecified (x). The first step is to randomly assign values to the unspecified PIs. To differentiate the randomly assigned values from the generated values the latter ones are printed in bold letters.

The normal way of fault dropping by a single pattern fault simulation is shown on the left side of Fig. 6. The two possibilities of pattern parallel fault dropping used in CONTEST are shown on the right side. First we have the pattern parallel fault dropping with random patterns. In this case bit 0 represents the generated test pattern and the bits 1 to 31 are filled up with random patterns. In the second case the generated test pattern (bit 0), six patterns based upon Hamming distance one variation of the generated PI assignments (bit 1 - 6), 15 patterns based upon Hamming distance two variation (bit 7 - 21) and with the remaining bits ten patterns based upon Hamming distance three variation are simulated. The variation is marked by underlining the concerned bits. The originally unspecified PI values are set randomly for each pattern independent from the Hamming distance variation of the generated PI values.

The pattern parallel fault dropping based upon Hamming distance variations of the generated PI values decreases the total CPU-time needed for test pattern generation as shown by the results presented in chapter five.

4 Overview of CONTEST

In the first step CONTEST reads a flattened gate level description generated from a hierarchical EDIF description or from an in house ASCII format. Primitive elements supported by CONTEST are unidirectional inputs and outputs, and all types of primitve gates like AND, NAND, OR, NOR, EXOR, EXNOR, inverting and non inverting buffers. Once the data structure is built the circuit graphs connectivity will be analized. In the ATPG system CONTEST single stuck-type faults are processed. Normally a fault list is read from a file. If no file exists a fault list is generated automatically using the relationships of fault equivalence and fault dominance.

Like other ATPG systems CONTEST uses controllability measures to guide the multiple backtracing. Three different controllability measures are possible:

1. Combinational zero/one controllability as determined in SCOAP [14].
2. A probabilistic controllability based on the results of a sampling simulation using a fast compiler-driven fault simulator.
3. Controllability derived from circuit levelizing.

In the next step either cone-oriented or conventional circuit-oriented test pattern generation will be executed. During test pattern generation faults are selected one at a time and after a test pattern is generated a pattern parallel fault dropping with random patterns or patterns based on Hamming distance variation is carried out. If there still remain aborted faults a second run using dynamic implications and a backtrack limit increased by a factor of ten will be performed.

After test pattern generation the test pattern set size will be reduced by a reverse order fault simulation of the generated test patterns. In such a manner the test set size will be reduced in the order of 40 percent. In the end, a list of redundant and if necessary aborted faults is generated and an execution report is printed.

5 Experimental Results

CONTEST is designed as a fast ATPG tool for very large combinational circuits. It has been implemented in C and runs on an Apollo DN 4500 workstation. To allow a realistic comparison between the circuit-oriented and cone-oriented test pattern generation CONTEST has been designed to perform both. The presented results are obtained from the ISCAS'85 benchmark circuits [15] and the five largest ISCAS'89 benchmark circuits [16]. The flip-flops in the five latter circuits are assumed to be connected to a Scan Path. Table 1 summarizes the characteristics of these circuits.

Circuit	Nodes	Inputs	Outputs	Faults
c432	446	36	7	463
c499	532	41	32	708
c880	934	60	26	799
c1355	1.368	41	32	1.316
c1908	1.650	33	25	1.684
c2670	2.346	157	64	2.334
c3540	2.934	50	22	3.020
c5315	4.780	178	123	4.903
c6288	6.312	32	31	6.294
c7552	6.783	206	107	6.793
s15207	14.186	700	790	8.622
s15850	16.276	611	684	10.283
s35932	39.648	1.763	2.048	34.144
s38417	39.442	1.664	1.742	27.582
s38584	37.936	1.464	1.730	32.126

Table 1: Characteristics of the benchmark circuits

The following experiments have been carried out (backtrack limit = 3):

1) Conventional circuit-oriented test pattern generation and pattern parallel fault dropping using Hamming distance variations.
2) Cone-oriented test pattern generation and pattern parallel fault dropping using Hamming distance variations.
3) Cone-oriented test pattern generation and pattern parallel fault dropping using random patterns.

Table 2 and 3 show the results of the first two experiments. The performance comparison will be depicted in terms of five parameters: the number of aborted faults (aborted), the number of identified redundancies (redundant), the CPU-time used for DTPG (t_{gen}) and for fault simulation (t_{sim}) and the total number of backtracks (backtracks) during DTPG.

Circuit	aborted	redundant	tgen[s]	tsim[s]	backtracks
c432	-	4	2	1	10
c499	-	8	5	2	11
c880	-	-	<1	1	0
c1355	-	8	15	5	21
c1908	-	9	10	7	8
c2670	-	115	83	8	159
c3540	-	135	13	19	1
c5315	-	59	2	14	2
c6288	-	36	12	77	3
c7552	-	131	244	40	466
s13207	-	149	261	198	284
s15850	-	384	179	244	46
s35932	-	3.728	87	333	0
s38417	-	161	122	2.307	92
s38584	-	1.345	105	1.205	71

Table 2: Results of the cone-oriented test pattern generation

Circuit	aborted	redundant	tgen[s]	tsim[s]	backtracks
c432	-	4	4	1	10
c499	-	8	5	2	2
c880	-	-	1	1	0
c1355	-	8	27	6	29
c1908	-	9	18	7	13
c2670	4	111	1.830	10	283
c3540	-	135	31	18	32
c5315	-	59	23	15	65
c6288	-	36	31	65	6
c7552	-	131	1.180	43	425
s13207	-	149	289	213	101
s15850	-	384	1.228	227	324
s35932	-	3.728	2.496	362	0
s38417	-	161	522	2.271	298
s38584	-	1.345	1.620	1.093	772

Table 3: Results of the circuit-oriented test pattern generation

Circuit	Random Patterns		HD Patterns	
	tgen[s]	tsim[s]	tgen[s]	tsim[s]
c432	3	1	2	1
c499	9	1	5	2
c880	1	2	<1	1
c1355	24	6	15	5
c1908	19	9	10	7
c2670	90	14	83	8
c3540	30	29	13	19
c5315	3	15	2	14
c6288	60	73	12	77
c7552	760	60	244	40
s13207	293	410	261	198
s15850	203	579	179	244
s35932	80	1.367	87	333
s38417	136	4.927	122	2.307
s38584	146	4.521	105	1.205

Table 4: Comparison between random pattern fault dropping and fault dropping based on Hamming distance variations

The results clearly underline the superiority of the cone-oriented approach. For each circuit a complete test set for all detectable stuck-at faults and a 100 percent redundancy identification is achieved (table 2). With respect to the number of aborted faults and identified redundancies there is no significant difference between cone- and circuit-oriented ATPG. The same holds true for the simulation time which lies in the same order for the two approaches. But for 13 of 15 circuits the tests could be generated considerably faster than for the unpartitioned circuit. The total amount of time for DTPG (t_{gen}) is reduced by a factor of 8.

The comparison between random pattern fault dropping and fault dropping based on Hamming distance variations is shown in table 4. In all circuits the fault dropping using patterns based on Hamming distance variations results in a remarkable reduction of the fault simulation time. Especially for the larger circuits (> 10.000 nodes) the Hamming distance variation of generated test patterns has a favourable effect. For these circuits an acceleration of the fault simulation by a factor of 3 is achieved.

6 Conclusion

CONTEST a fast ATPG tool for large combinational circuits has been presented. The performance improvement is based on four major ideas. Cone-oriented circuit partitioning is shown to reduce the circuit complexity thus accelerating ATPG. The propagation graph is introduced as a dynamic data structure keeping track of all paths from the fault location to a primary output. Dynamic implications used as part of the multiple backtrace procedure to reduce the number of contradictory node assignments. Pattern parallel fault dropping based on Hamming distance variations of generated test patterns reduces the simulation time.

For all examined ISCAS benchmark circuits containing up to 40.000 nodes complete test sets for all detectable stuck-at faults have been generated. Additionally a 100 percent redundancy identification has been achieved. Compared to the circuit-oriented test pattern generation the cone-oriented approach as implemented in CONTEST is by an average factor of 8 faster. The new fault dropping technique based on Hamming distance variations of generated test patterns results in a reduction of fault simulation time by an average factor of 3 compared to random pattern simulation.

Further current work is concerned with the modification of the ATPG-system as test pattern generator for delay faults. Additionally research is done on modifying CONTEST as a diagnostic pattern generator and as a tool capable of generating short test sequences for electron beam tester supported fault diagnosis.

References

[1] Williams, M.J.Y., Angell, J.B.: "Enhancing Testability of Large Scale Integrated Circuits via Test Points and Additional Logic", IEEE Trans. Comput., C-22 (1), pp. 46-60, 1973

[2] Kobayashi, T., Matsue, T., Shiba, H.: "Flip-flop Circuit with FLT Capability", Proc. IECEO, p. 692, 1968

[3] Eichelberger, E.B., Williams, T.W.: "A Logic Design Structure for LSI Testability", Proc. 14th Design Automation Conference, pp. 462-468, 1977

[4] Roth, J.P.: "Diagnosis of Automata Failures: A calculus and a method", IBM Journal Res. Dev. 10, pp. 278-281, 1966

[5] Goel, P.: "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic", IEEE Trans. Comput., Vol. C-30 (3), pp. 215-222, 1981

[6] Fujiwara, H., Shimono, T.: "On the Acceleration of Test Generation Algorithms", Proc. 13th Int. Symp. Fault-Tolerant Computing, pp. 98-105, 1983

[7] Kirkland, T., Mercer, M.R.: "A Topological Search Algorithm for ATPG", Proc. 24th Design Automation Conference, pp. 502-508, 1987

[8] Schulz, M.H., Auth, E.: "Advanced Automatic Test Pattern Generation and Redundancy Identification Techniques", Proc. 18th Int. Symp. Fault-Tolerant Computing, pp. 30-35, 1988

[9] Grüning, T., Mahlstedt, U., Daehn, W., Özcan, C.: "Accelerated Test Pattern Generation by Cone-Oriented Circuit Partitioning", IEEE Proc. 1th European Design Automation Conference, pp. 418-421, 1990

[10] Tarjan, R.: "Finding Dominators in Directed Graphs", SIAM Journal of Computing, Vol. 3, pp. 62-89, 1974

[11] Church, A.: "Introduction to Mathematical Logic", Princeton, New Jersey, Princeton University Press, 1956

[12] Muth, P.: "A Nine-Valued Logic Model for Test Generation", IEEE Trans. Comput., Vol. C-25, pp. 630-636, 1976

[13] Köppe, S., Starke, C.W.: "Logiksimulation komplexer Schaltungen für sehr große Testlängen", NTG-Fachberichte Großintegration, pp. 73-80, März 1985

[14] Goldstein, L.H.: "Controllability/Observability Analysis of Digital Circuits", IEEE Trans. on Circuits and Systems, Vol. CAS-26, No. 9, pp. 685-693, Sept. 1979

[15] Brglez, F., Fujiwara, H.: "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran", special session on ATPG and fault Simulation, Proc. 1985 IEEE Int. Symp. on Circuits and Systems, Kyoto (Japan), 1985

[16] Brglez, F., Bryan, D., Kozminski, K.: "Combinational Profiles of Sequential Benchmark Circuits", IEEE Int. Symp. on Circuits and Systems, 1989