

FOGBUSTER: An Efficient Algorithm for Sequential Test Generation

U. Gläser, H. T. Vierhaus

The German National Research Center for Computer Science (GMD)

System Design Technology Institute

Schloß Birlinghoven

D53757 St. Augustin

email: glaeser@gmd.de

Tel: (+49)2241-14-2875, FAX: (+49)2241-14-2242

Abstract:

Automatic test pattern generation yielding high fault coverage for CMOS circuits has received a wide attention in industry and academia for a long time. Scan techniques were used to break down the sequential ATPG problem to combinational test generation. As the overhead necessary for scan design can not be spent for all circuits, sequential test generation techniques gained important. In this paper the FOGBUSTER (FORward propaGation Backward jUstification Sequential Test genERation) algorithm is described and experimental results for the ISCAS '89 benchmark circuits are shown. FOGBUSTER is a complete algorithm which makes use of a forward propagation technique which is more efficient in general than the backward propagation technique used in the well known BACK-algorithm.

1. Introduction

Automatic test generation for combinational logic based on the FAN algorithm [FuSh83], relying on the D-algorithm [Roth66] has reached a high level of maturity. FAN has also been modified for test generation in synchronous sequential circuits presented in [ChBA88], [SuAu89], [ChCh89], [GoKa91], [NiPa91]. In earlier papers ([GlHü92], [GlVi92]) we have presented a test generation technique that combines the accuracy of switch-level test generation with the efficiency of gate-level test generation. As combinational transition fault test generation is not easily applicable to sequential circuits in combination with conventional scan design, sequential test generation is one solution to test generation for transition fault models. A second solution could be the use of modified scan techniques requiring an additional hardware effort.

Recently the detection of defects beyond functional faults by methods such as built-in overcurrent measurements (Iddq-testing) [MaNi88], [StMa91] has received considerable attention. Despite their potential coverage of transistor faults and bridging faults, these methods are static by nature and therefore they are a

complement to rather than a replacement for dynamic testing. Thus test generation for overcurrent measurement is optionally included in our tools.

The work introduced here is aimed at the generation of efficient test sets for conventional voltage-based tests as well as for Iddq tests.

A sequential test pattern generation approach is presented in this paper. It is based on the new FOGBUSTER-algorithm. In contrast to the BACK-algorithm [ChBA88] which was built on basic theoretical work [Marl78], [MaWu85], [Marl86], [MaDe87], [KeMa87], FOGBUSTER uses a forward propagation and backward justification technique, which is in general more efficient than the exclusive reverse time processing BACK uses.

2. The FOGBUSTER-algorithm

The FOGBUSTER algorithm (FORward propaGation Backward jUstification Sequential Test genERator) was developed to overcome deficiencies of the state-of-the-art test generators HITEC [NiPa91] and GENTEST [ChCh89]. This section is organized as follows: Synchronous circuits that can be handled by our ATPG are characterized in section 2.1. The main differences between FOGBUSTER and the BACK-algorithm are stated in section 2.2. This is followed by a description of FOGBUSTER (section 2.3) and a detailed description of the forward propagation technique in section 2.4. In section 2.5 the treatment of the over specification problem is presented and finally in section 2.6 and section 2.7 the test generator SEMILET using the FOGBUSTER algorithm is described. Experiments with the ISCAS '89 benchmarks [BrBr89] are presented in section 2.8.

2.1. Circuits

FOGBUSTER and SEMILET can handle a superset of synchronous sequential circuits that HITEC [NiPa91] and GENTEST [ChCh89] accept. These circuits have to meet the following characteristics:

- All feedback loops include at least one storage element.
- All storage elements have a clock input which is connected to one global clock signal, one data input and one non-inverting data output. We assume that the storage elements are master-slave flip-flops, or edge-triggered D-flip-flops.
- Global set and/or reset signals which affect all storage elements are allowed. The test generator can take advantage of them.
- The circuit may consist of primitive- (AND, NAND, OR, NOR) and combinational complex gates.

Basically FOGBUSTER can deal with finite state machines with just one clock depicted in figure 1.

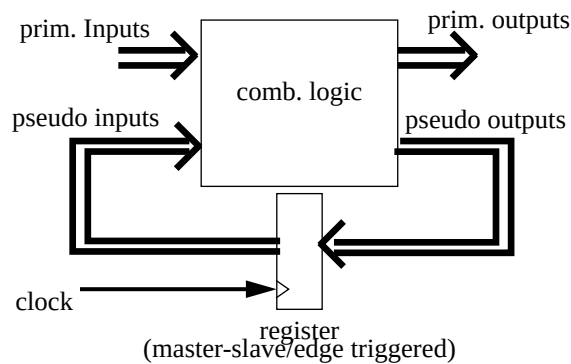


Figure 1: Finite state machine model

Each circuit consists of a number of primary inputs and a number of primary outputs. In addition the circuits have pseudo inputs, the outputs of the sequential elements, and pseudo outputs, the inputs of the sequential elements. The sequential elements themselves can be handled as a register. They store the actual state of the circuit.

In general it is not possible to generate a testpattern directly from such a circuit because one special state is needed for the fault injection and a synchronizing sequence to this state at least has length 1. It is also possible that more than one clock cycle is needed to propagate the fault effect to the primary outputs of the circuit. For instance, a binary counter may need 2^n clock cycles for the propagation path depending on the fault location, where n is the number of flip-flops in the counter.

By giving as many time frames to the circuit as needed for the maximal test length, the circuits could be treated by a multifault combinational test generator as shown in [ChBA88]. This method is impractical because of the memory space required for copying circuit and register. Thus a window is used scrolling over the time frames. In most of test generation systems [GoKa91] [SuAu89] [ChCh89] the window size is one time frame. The only test generator operating with larger window size is HITEC [NiPa91].

2.2. The general approach in comparison with other algorithms

As stated in the introduction, the most common approach in sequential test generation is based on the BACK-algorithm [ChBA88]. Starting from a target fault, BACK selects first a primary output, where the fault will be visible a number of time frames later. This selection is supported by a drivability measure, an additional approach to testability analysis [SuAu89], [GoKa91]. After selecting a primary output, the fault propagation path starting from the selected primary output backwards to the fault location is computed. Finally, the needed state at the fault location is generated in the backward direction.

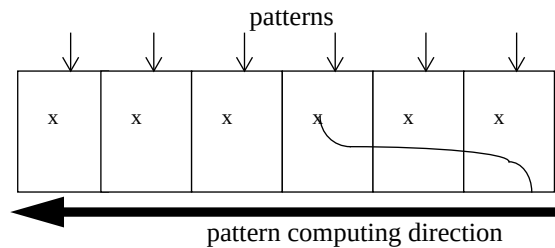


Figure 2: The BACK-algorithm

In figure 2 a diagram for the BACK algorithm is shown. It is visible that the pattern computing direction is reverse, e.g., the values of the time frame, where the fault effect occurs at a circuit output, are computed first, and the values for the time frame with the initial state are computed last. Thus the fault propagation path and the fault justification are computed by reverse time processing.

Nearly all test pattern generators at the gate-level for sequential circuits [GoKa91], [SuAu89], [SeAb90], [ChAb90], [KeMa87], [ChCh89], [MaWu85] make use of a technique based on the BACK-algorithm. The only known approach which uses a different technique is HITEC [NiPa91]. In HITEC Niermann and Patel use a forward propagation backward justification technique which means that the propagation is performed by forward processing and the line justification is done using reverse time processing. This technique is illustrated in figure 3.

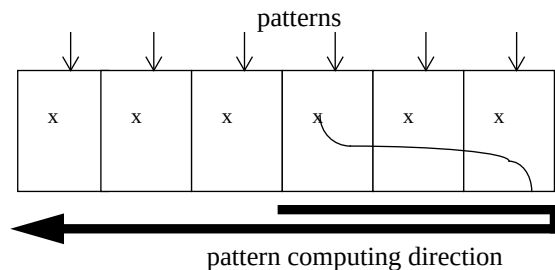


Figure 3: The forward propagation backward justification technique

The main advantages of forward propagation backward justification over the BACK algorithm are:

- No drivability is needed for the test generation process, i.e., computing time is saved.
- In the BACK algorithm the selection of the output where the fault is finally visible requires a large search space, e.g., the initial decision tree is relatively large. Thus, if it is impossible to propagate the fault effect to this output, a backtrack to a different output has to be performed.
- It is possible to use efficient forward propagation techniques (unique sensitisation, dominators) which are well known in combinational test generation; in the reverse time processing of the BACK algorithm, these techniques are of no use.

Characteristic for HITEC is that it selects the best observable signal for the fault propagation leaving all others in a stack for backtracking possibility. The main disadvantage of HITEC is:

- In HITEC a large decision tree is built during the propagation by selecting one signal for driving the fault effect to a primary output (targeted D-front). Thus many backtracks requiring much CPU time are needed for redundancy identification.

A feature of HITEC is that this program can handle more than one time frame at the same time during propagation. This results into reduced CPU-time effort, because value assignment in later time frames can be performed. The main disadvantage of this technique is that much memory space is required for multiple storing of time frames.

To overcome both the disadvantages of BACK and HITEC, an efficient test generator using forward propagation backward justification technique (FOGBUSTER) was implemented. FOGBUSTER and the corresponding test generation tool SEMILET (SEquential MIXed LEvel Testgenerator) are described in the following sections.

2.3. The test generation technique

In this section the different phases of FOGBUSTER (FOrward propaGation Backward jUstification SEquential TESt geneRation) are described. The algorithm proceeds in three main phases of computation:

1. forward propagation phase
2. propagation justification phase
3. initialization phase (or justification phase)

In the first phase, the fault is transported to at least one primary output of the circuit. Efficient heuristics for fault propagation are used in this phase, for instance unique sensitisation [FuSh83] with detection of dynamic dominators. In the second phase, the propagation path is justified, e.g., unjustified signals of the first phase, which are at the pseudo outputs of the circuit, are justified. In the initialization phase, the initial state of the time frame of fault occurrence is confirmed,

e.g., a synchronizing sequence for this state is computed.

If a fault is hard to detect the computing time for finding a test pattern for this fault is too large. To keep the computing time low, the maximal number of backtracks for one fault can be limited by the user. Hard faults are very often detected by a test pattern for a different fault. Therefore, after a test pattern for any fault is found, fault simulation for all remaining faults (which are not detected by any previous test pattern) of the fault list is performed.

2.4. Fault propagation and propagation justification

The fault propagation phase and the propagation justification phase are used to propagate a fault from the fault location to a primary output. The object is to find an input-sequence resulting in a difference at least at one output of the good- and the faulty machines.

In each time frame the actual state is set to the pseudo inputs of the circuit. The non-empty set of signals which are on the propagation path are added to the D-front [FuSh83] which is at any time built of the "end point signals" of all pathes transporting the fault effect in the multi-fault FAN-algorithm. The object is to propagate the signals of the D-front until one output is included in the D-front, e.g., until the fault effect is visible at one output. The fault location is marked and added to the D-front, when a different value in the good machine is computed here. The multi-fault FAN terminates:

- If the D-front cannot be propagated through this time frame. A backtrack to the previous time frame has to be performed.
- If the D-front reaches a primary output. The propagation is finished. The propagation justification phase is started.
- If the D-front is completely propagated to pseudo outputs (also partly dissolving of the D-front is possible). This is done because of two reasons. On one hand, for every pseudo input which is not included in the D-front by entering the time frame, the good and the faulty machine are equal. On the other hand, after reaching a pseudo output of the circuit, a primary output and thus a shorter test sequence could possibly be reached. If the propagation of the actual time frame would be stopped when reaching a pseudo output, also the completeness would be lost. This will be illustrated later using figures 5 and 6.

In each time frame some pseudo inputs may have initial values X, because values here are not needed for fault propagation in the previous time frame. If these signals have value requirements in the actual time frame, they are marked unjustified and prepared for justifying in the propagation justification phase. The latter has to be computed using reverse time processing.

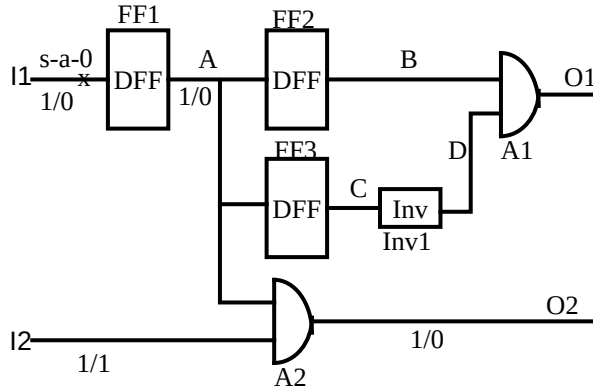


Figure 5: Example circuit: full D-front propagation

In figure 5 a cycle free circuit is shown. I1 and I2 are the prime inputs of the circuit and O1 and O2 are the prime outputs respectively. Using the FOGBUSTER algorithm, two time frames are needed to generate a test for the stuck-at-0 fault at signal I1. In the first time frame the primary input I1 is set to "1" to initialize the fault and thus propagate the fault to the D-flip-flop FF1. In the second time frame, the values of all pseudo outputs of the first time frame of the circuit are assigned to the pseudo inputs, thus the signal A is set to "1/0", that means "1" in the good circuit and "0" in the faulty circuit. By using a simple implication the D-front consists of three signals, the fanout branches of A. Two of them are pseudo outputs of the circuit and the third signal of the D-front is at gate A2. For the D-front has to be fully propagated, it is not possible to step to the next time frame immediately. First an optional assignment has to be performed at signal I2 to propagate the D-front completely. In our example, I2 is set to "1", and by performing an implication the D-front changes. Now O2 is in the D-front and thus the fault effect is propagated to a primary output of the circuit and the test pattern is generated.

Note that the assignment of signal I2 to "1" is an optional assignment. It is also possible that I2 was assigned to "0" instead. This assignment can be reverted by backtracking if necessary.

If the D-front was not completely propagated, the completeness of test generation would be lost. The meaning of a complete algorithm is that for every testable fault a test pattern is found by the algorithm.

In figure 6 the effect of non-complete D-front propagation is shown using the same example circuit. When the D-front reaches FF2 and FF3, a direct step into time frame 3 is executed, with signals B and C in the D-front. After performing a simple implication, the D-front is empty, and thus a backtrack has to be performed. As there was no optional assignment in the test generation process up to here, there is no backtracking possibility, and the fault is marked redundant although, as seen in figure 5, it is testable.

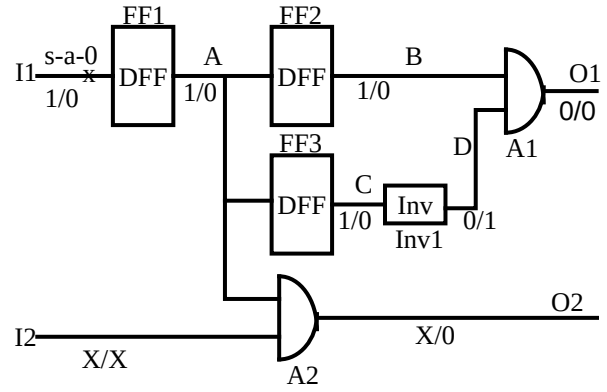


Figure 6: Example circuit: non-complete D-front propagation

2.5. The over-specification problem in sequential test generation

A main problem of test generation for sequential circuits is the over specification problem. This problem results from mapping the backtracking technique of combinational test generation to sequential test generation. At worst case, this problem may result in loosing completeness in test generation. Solutions of the problem are the consequent value model described in [GoKa91] and the use of a ternary decision tree described in [ChMa93], which both rely on increasing the logic used in the test generation. For completeness reason our algorithm uses the method described in [ChMa93].

2.6. Detection of state repetitions in test generation

To get a relatively short test sequence, state repetitions in the test sequence have to be avoided.

Theorem ([Micz83]): If a test sequence leading the good and the faulty machines to states $q_1, \dots, q_i, \dots, q_j, \dots, q_r$, $i, j, r \in \mathbb{N}$, $1 < i < j < r$, with $q_i = q_j$ is a test for fault F, the test sequence leading the good- and faulty machines to $q_1, \dots, q_i, q_{j+1}, \dots, q_r$ is also a test for fault F.

A second theorem of [Micz83] is that a synchronizing sequence leading to a defined state (all storage elements store 0 or 1) of all storage elements is upper bounded by $3^n - 2^n$, n is the number of storage elements in the circuit, assuming that there is no global reset possibility. The reason is that a 3-valued logic is used starting from the undefined state. As there are n storage elements, the number of states in the circuit is upper bounded by 3^n . Since no state repetition is allowed and since the number of states, where all storage elements have a defined state, is 2^n , the storage elements have a defined state after applying an input sequence with length at most $3^n - 2^n$. A second result is that all reachable states of the circuit can be reached by a sequence which has at most a length of 3^n .

To avoid long test sequences with state repetitions, SEMILET checks during propagation and in the initialization phases whether the actual state has occurred a number of time frames earlier in the test generation. If this is true, a backtrack to the previous time frame is performed and test generation is continued without loosing the completeness of the algorithm.

2.7. The use of global set- and reset-signals in ATPG

If the circuits consist of global set- and reset-signals, test generation can take advantage of this. A global set signal effects that every flip-flop in the circuit is set to 1, and a global reset signal effects that every flip-flop is set to value 0. In the initialization phase of the algorithm it is possible to make use of global set- and reset signals:

- A set can be performed, if in the current state in ATPG there is no 0-requirement.
- A reset can be performed, if in the current state in ATPG there is no 1-requirement.

SEMILET can optionally use these features. The results with ISCAS-benchmark circuits in section 2.8. Table 3 shows that the use of global set- and/or reset-signals results in increased fault coverage and/or decreased CPU-time effort as expected.

2.8. Experimental results

At GMD we have developed SEMILET for test generation for synchronous sequential circuits. SEMILET makes use of the FOGBUSTER-algorithm and has mixed level capabilities like MILEF. In the actual state of implementation, SEMILET can handle 3 fault models:

- stuck-at with overcurrent detection (mode 1)
- stuck-at with state propagation (mode 2)

SEMILET is programmed in C++ with about 30,000 lines of code. It is prepared for sequential test generation with two levels of hierarchy, the switch-level and the gate-level (see also [GIHu92]). The time frame model is applied only at gate-level.

The overcurrent techniques are reported in [MaNi88] and in [StMa91]. Generation of test patterns for overcurrent techniques is in general easier in comparison with the test generation for voltage based fault models.

Table 1 shows experimental results of SEMILET with fault model mode 1 in comparison with results of GENTEST [ChCh89] for some ISCAS '89 benchmark [BrBr89] circuits. The times were measured on a SUN 4/260.

Looking at results of table 1 it is visible that SEMILET has almost the same fault coverage and performance in comparison with GENTEST for IDDQ models. For some of the circuits (s344, s349, s838) SEMILET is a bit faster and for other circuits (s400, s713) SEMILET is a bit slower than GENTEST. This result was

theoretically expected, because only justification is used for computing test patterns for overcurrent related fault models since no fault propagation is required. Fault justification in FOGBUSTER is equal to fault justification in BACK.

Table 1. Performance of SEMILET and GENTEST in comparison (IDDQ test mode 1):

Circuit	GENTEST		SEMILET	
	efficiency	CPU-time/s	efficiency	CPU-time/s
s208	100%	3	100%	2
s298	100%	6	100%	2
s344	100%	12	100%	1
s349	100%	16	100%	1
s400	100%	120	100%	217
s420	100%	17	100%	5
s641	100%	6	100%	6
s713	100%	22	100%	23
s838	100%	71	100%	42
s953	100%	2	100%	1
s1196	100%	7	100%	6
s1238	100%	7	100%	7
s5378	98.4%	27941 *	96.2%	21113

*: Results were computed on a Convex C-200

Table 2 shows the comparison of GENTEST, HITEC and SEMILET for the stuck-at fault model with state propagation. A comparison of BACK, HITEC and FOGBUSTER is possible using these results.

Table 2. Performance of SEMILET, HITEC and GENTEST in comparison (stuck-at mode 2):

Circuit	GENTEST		HITEC		SEMILET	
	fault coverage	CPU-time/s	fault coverage	CPU-time/s	fault coverage	CPU-time/s
s208	63.7%	15	63.7%	24	63.7%	7
s298	85.7%	2099	*	*	85.7%	355
s344	96.2%	334	95.9%	4775	96.2%	640
s349	95.7%	489	95.7%	3129	95.7%	625
s386	81.8%	878	81.8%	62	81.8%	80
s420	41.6%	478	41.6%	2701	41.6%	280
s510	0%	1	0%	2	0%	0 **
s641	86.5%	11	86.5%	737	86.5%	15
s713	81.4%	438	81.9%	84	81.9%	17
s1196	99.8%	38	99.8%	41	99.8%	33
s1238	94.7%	61	94.7%	183	94.7%	58
s5378	74.0%	10 ⁵ ***	*	*	74.0%	22605
s9234	0.3%	530	0.3%	63	0.3%	10

*: No results known for this circuit

**: Redundancies were detected in the preprocess

***: Results were computed on a Convex C-200

Table 2 shows the stuck-at performance of SEMILET in comparison with HITEC and GENTEST. It is shown that the fault coverage of all approaches is equal (exception: for s344 the fault coverage of HITEC is lower). In most cases the test generation time required for SEMILET was smaller in comparison with HITEC (except s386). The test generation times for GENTEST

were sometimes a bit smaller (s344, s349, s641) but sometimes much larger (s298, s386, s713, s5378) than the SEMILET times. In many cases it was measured in the experiments that, if one of the test pattern generators HITEC or GENTEST is a bit faster than SEMILET, the other test pattern generator is much slower (example: s344, s349, s386,...). Thus SEMILET has much better results than HITEC and GENTEST in the average.

Table 3. Performance of SEMILET with and without possibilities of global setting and resetting (stuck-at patterns) mode 2:

circuit	without possibility of setting and resetting		with possibility of setting and resetting	
	fault coverage	CPU-time/s	fault coverage	CPU-time/s
s208	63.7%	7	78.1%	42
s298	85.7%	355	87.7%	264
s344	96.2%	640	99.4%	140
s349	95.7%	625	96.2%	285
s386	81.8%	80	81.8%	52
s420	41.6%	280	58.8%	592
s510	0%	0 **	100%	262
s641	86.5%	15	94.4%	137
s713	81.9%	17	87.6%	185
s1196	99.8%	33	99.8%	22
s1238	94.7%	58	94.7%	32
s5378	74.0%	22605	74.9%	15262
s9234	0.3%	10	6.3%	56630

Table 3 shows that the use of global set and reset signals results in increased fault coverage at reduced testing time. The fault coverage increased for instance for s510 and s420. The testing time was reduced for instance for s344 and s5378.

3. Conclusion

This paper presents a new approach to sequential test generation. A new efficient algorithm called FOGBUSTER for the forward propagation backward initialization technique handles synchronous sequential circuits. A 4-valued logic and a two-staged backtracking mechanism is used to handle the over-specification problem and to achieve completeness. Experimental results for the ISCAS '85 and ISCAS '89 benchmark circuits are encouraging.

4. References

- [BrBr89] F. Brglez, F. Bryant, D. Kozminski: *Combinational Profiles of Sequential Benchmark Circuits*, Proc. 1989 Int. Symp. Circ. and Systems
- [ChAb90] C. H. Chen, J. A. Abraham: *Mixed-Level Sequential Test Generation Using a Nine-Valued Relaxation Algorithm*, Proc. ICCAD '90, pp. 230-233
- [ChBA88] W. Cheng: *The BACK-Algorithm for Sequential Test Generation*, Proc. ICCD '88, pp. 66-69
- [ChCh89] W. Cheng, T. J. Chakraborty: *GENTEST An Automatic Test-Generation System for Sequential Circuits*, IEEE Computer '89, pp. 43-49
- [ChMa93] K. T. Cheng, T. Ma: *On the over-specification problem in sequential ATPG algorithms*, IEEE Trans. Computer aided design, october 1993, pp. 1599-1604
- [ChSP88] W. T. Cheng: *Split Circuit Model for Test Generation*, Proc. 25. DAC 1988, pp. 96-101
- [FuSh83] H. Fujiwara, T. Shimono: *On the Acceleration of Test Generation Algorithms*, IEEE Trans. on Computers (C-32), pp. 1137-1144, 1983
- [GlHü92] U. Glaeser, U. Hübner, H. T. Vierhaus: *Mixed level hierarchical test generation for transition faults and overcurrent related defects*, Proc. ITC '92, pp. 21-29
- [GIVi92] U. Glaeser, H. T. Vierhaus: *MILEF: An efficient approach to mixed level automatic test pattern generation*, Proc. EURODAC '92, pp. 318-321
- [Gold79] L. H. Goldstein: *Controllability/Observability Measure for Logic Testability*, IEEE Trans. Circuits and Systems, Vol. CAS-26, pp. 685-693, 1979
- [GoKa91] N. Gouders, R. Kaibel: *Advanced Techniques for Sequential Test Generation*, Proc. 2.nd European Test Conference, Munich, 1991
- [KeMa87] H. Keung, T. Ma, et al.: *Test Generation for Sequential Finite State Machines*, Proc. ICCAD '87, pp.288-291
- [MaDe87] T. Ma, S. Devadas, A. R. Newton, A. Sangiovanni-Vincentelli: *Test generation for sequential circuits*, IEEE Transactions Computer aided design, 1888 pp. 1081-1093
- [MaNi88] W. Maly, P. Nigh: *Built-in Current Testing-a Feasibility Study*, Proc. IEEE ICCAD'88, pp. 340-343, 1988
- [Marl78] R. Marlett: *EBT: A Comprehensive Test Generation Technique for Highly Sequential Circuits*, Proc. DAC '78, pp. 335-339
- [Marl86] R. Marlett: *An Effective Test Generation System for Sequential Circuits*, Proc. DAC '86, pp. 250-256
- [MaWu85] S. Mallela, S. Wu: *A Sequential Circuit Test Generation System*, Proc. ITC '85, pp. 57-61
- [Micz83] A. Miczo: *The Sequential ATPG: A Theoretical Limit*, Proc. DAC '83, pp. 143-147
- [NiPa91] T. Niermann, J. H. Patel: *HITEC: A test generation package for sequential circuits*, Proc. EDAC '92, pp. 214-218
- [Roth66] J. P. Roth: *Diagnosis of Automata Failures: A Calculus and a Method*, IBM Journal 1966, pp. 278-291
- [SeAb90] C.-H. Shen and J. A. Abraham: *Mixed Level Sequential Test Generation Using a Nine Valued Relaxation Algorithm*, Proc. ICCAD-90, pp.230-235
- [StMa91] T. Storey, W. Maly, J. Andrews, M. Miske: *Comparing Stuck Fault and Current Testing via CMOS Chip Test*, Proc. 2nd European Test Conf., Munich, 1991, pp. 149-156
- [SuAu89] M. H. Schulz, E. Auth: *Essential: An Efficient Self-Learning Test Pattern Generation Algorithm for Sequential Circuits*, Proc. IEEE Int. Test Conf. 1989, pp. 28-37