Gate Delay Fault Test Generation for Non-Scan Circuits

G. van Brakel*, U. Gläser**, H.G. Kerkhoff* and H.T. Vierhaus**

*MESA Research Institute,
University of Twente,
P.O.-box 217, 7500 AE Enschede,
the Netherlands

**GMD - The German National Research
Center for Computer Science,
Schloß Birlinghoven, 53757 St. Augustin,
Germany

abstract:

This article presents a technique for the extension of delay fault test pattern generation to synchronous sequential circuits without making use of scan techniques. The technique relies on the coupling of TDgen, a robust combinational test pattern generator for delay faults, and SEMILET, a sequential test pattern generator for several static fault models. The approach uses a forward propagation - backward justification technique: The test pattern generation is started at the fault location, and after successful "local" test generation fault effect propagation is performed and finally a synchronising sequence to the required state is computed. The algorithm is complete for a robust gate delay fault model, which means that for every testable fault a test will be generated, assuming sufficient time. Experimental results for the ISCAS'89 benchmarks are presented in this paper.

1: Introduction

Requirements upon VLSI designs are continuously increasing towards faster and larger circuits, leading to area and timing optimized designs, and raising demands for testing. Testing should be thorough, to confirm high specifications, but should not require area expensive Design for Testability circuitry. This calls for test pattern generation for more realistic fault models as the widely used stuck at fault model, like the delay fault model. It also calls for test pattern generation for sequential circuits without scan paths. Both of these areas have been discussed in recent papers, describing each however only one aspect of the problem: either it gives an approach

- to delay fault testing only in combinational [1, 2, 3, 4, 5, 6, 7, 8, 9] or scan sequential circuits [10, 11, 12, 13],
- or to testing of sequential circuits [14, 15, 16, 17, 18], but only for static fault models.

The few literature available on a combined approach [19] is restricted to small, resetable synthesised circuits, whereas in this paper a combined approach to test pattern generation for delay faults in general synchronous sequential circuits is presented. The presented system consists of a dedicated test pattern generator for delay faults in combinational blocks of sequential circuits, which is tightly coupled to a dedicated test pattern generator for static faults in sequential circuits that handles the sequential propagation and initialisation. This highly integrated approach results in effective test pattern generation for delay faults, yet without losing the guarantee to completeness.

The general approach is presented in section two. Section three describes the local test generation technique of TDgen. Global propagation and synchronization are described in section four. Section five describes the fault simulation, and experimental results are presented in section six. The paper ends with conclusions.

2: The general approach

Sequential circuits can be represented by a finite state machine model, as is shown in Figure 1. In such a circuit a delay fault exists if a signal is unable to propagate within the operational clock period from the Primary Inputs (PIs) or Pseudo Primary Inputs (PPIs) to the Primary Outputs (POs) or Pseudo Primary Outputs (PPOs). As illustrated in Figure 2 the test generation for such a circuit can be performed using the time frame model, while for all of the time frames except the test time frame a slow clock is used. For the test time frame a fast clock is used. Fast means that a delay fault with a realistic size can be detected as a faulty value at a PO, or as a faulty state of the circuit. Using such a clocking configuration the circuit behaves fault free during the initialization and propagation phases, and delay faults can only occur during the test time frame. If the delay fault results there in a faulty PPO the corresponding D or Dbar signal is propagated until a PO is reached. In the example in Figure 2 this propagation takes two time frames.

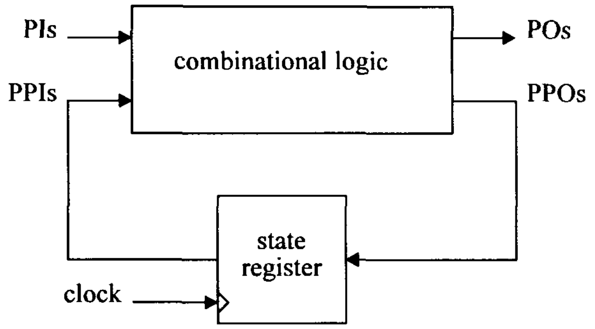

Figure 1: The finite state machine model

The test generation procedure is divided into the following steps:

1. Local test generation by selecting a fault and propagation of the fault effect to a PO or PPO.
2. Propagation of the fault effect to a PO, performed only if the fault effect was propagated to a PPO by the local test generator.
3. Justification of the test frames and the state required for propagation.
4. Initialization, i.e. finding a synchronizing sequence to the required state of the local ATPG.

Backtracking between these steps of the algorithm is possible achieving a complete approach, i.e. for every testable fault a test is generated assuming sufficient time is given. Note that after successful test generation fault simulation is performed for all faults that are up to this step untested.

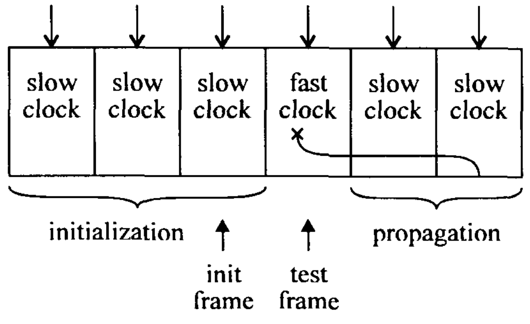

Figure 2: The time frame model

In our approach the test time frame and the time frame before, the initial time frame, are handled by a local test generator, TDgen. Propagation of the faulty value and synchronization of the required state of the initial time frame is performed by using the FOGBUSTER (forward propagation backward justification) technique and the test generator SEMILET. The local test generation is described in section 3. in detail. The FOGBUSTER technique is described later in section 4.

3: Local test pattern generation

For the delay fault test pattern generation in the combinational block of the sequential circuit the robust Gate Delay Fault model is assumed. Under this model each gate output and each fan out branch can contain a Slow-to-Rise (StR) and a Slow-to-Fall (StF) fault, that both need to be tested robustly. A fault is tested robustly if it is provoked by the appropriate transition, and that at an observable output (PO or PPO) the corresponding good circuit final value cannot be observed unless the correct final value is present at the fault location. This is illustrated in the algebra used by TDgen, that is shown in Table 1 and 2.

AND	0	1	R	F	0h	1h	R*c*	F*c*
0	0	0	0	0	0	0	0	0
1	0	1	R	F	0h	1h	R*c*	F*c*
R	0	R	R	0h	0h	R	R*c*	0h
F	0	F	0h	F	0h	F	0h	F
0h	0	0h	0h	0h	0h	0h	0h	0h
1h	0	1h	R	F	0h	1h	R*c*	0h
R*c*	0	R*c*	R*c*	0h	0h	R*c*	R*c*	0h
F*c*	0	F*c*	0h	F	0h	F	0h	F*c*

Table 1: Truth table for AND gate.

0	1	R	F	0h	1h	R*c*	F*c*
1	0	F	R	1h	0h	F*c*	R*c*

Table 2: Truth table for inverter

In this algebra the two time frames of the local test pattern generation are handled simultaneously. Basically a six-valued logic is used, consisting of the values 0, 1, R, F, 0h and 1h. The value 0 (1) represents a steady zero (one) value, without any hazard, during both time frames. The value R (F), meaning rising (falling) represents a zero (one) in the first time frame, and a one (zero) in the second time frame. The value 0h (1h) represents, similar to 0 (1), a zero (one) in both time frames, but in this case a hazard exists. A hazard is defined as the possibility that the signal may temporarily change it's value and change back, within the two time frames. For the ease of test pattern generation, two more values are added to the logic: R*c* and F*c*. These values represent rising and falling transitions, like R and F do, but they also carry the fault effect, like the values D and Dbar do in static test pattern generation. Table 1 shows the truth table for an AND gate for this logic. It shows in the rows and columns concerning the test carrying values R*c* and F*c* that tests generated using this logic will be robust. It is clear that R*c* propagates from the on path input to the output of the gate with any value on the off path input that is 1 in it's final value, but F*c* propagates only with a steady one or F*c* on the off path input. In all other cases the possibility exists that the fault effect becomes invisible, invalidating the robustness criterion. Note that an R*c* or F*c* value never emerges at an output of a gate if there wasn't already one or more of these values at the input. The only exception to this rule is made at the fault location itself, where an appropriate R or F value is converted into an R*c* or F*c*. This can however not be seen in Table 1. Table 2 shows the truth table for an inverter, that is not explained further. From these two truth tables the truth tables for the other primitive gates can be constructed using de Morgans rules. Another kind of truth table exists for the state register, that handles the correlation between the initial value of the PPO and the final value of the PPI.

During local test pattern generation for each gate a set of values is maintained that are possible for that gate [8][20]. Using these sets, and the truth tables for each gate, forward and backward implications can be made. The test pattern generator for delay faults TDgen is built around this logic system. TDgen implements a complete algorithm for the generation of robust test for delay faults in combinational circuits, or the combinational part of sequential circuits. In case of sequential circuits a part of the generated vectors has to be applied to the PPIs. For the final vector this part is the next state as produced by the initial vector. The initial state however has to be set by a sequence of initialisation vectors, generated by a test pattern generator for sequential circuits, as is described in section 4. Something similar is necessary for the observation of the fault effect: For some faults TDgen can only generate a test that requires the observation of the fault effect to be at a PPO instead of at a PO. In such a case the sequential test pattern generator has to generate a set of input vectors that make that PPO observable.

4: Sequential test generation

If the local test generation is successful the propagation part is called if the fault effect has reached a PPO. In that case the propagation of SEMILET is performed until a PO is reached by using forward time processing. The fault location is not needed to be known by SEMILET because a slow clock is assumed for the propagation phase and thus the fault does not occur in these time frames.

During the propagation phase to a PO it is possible that some values at PPIs are not justified directly. For this justification task the propagation justification phase is

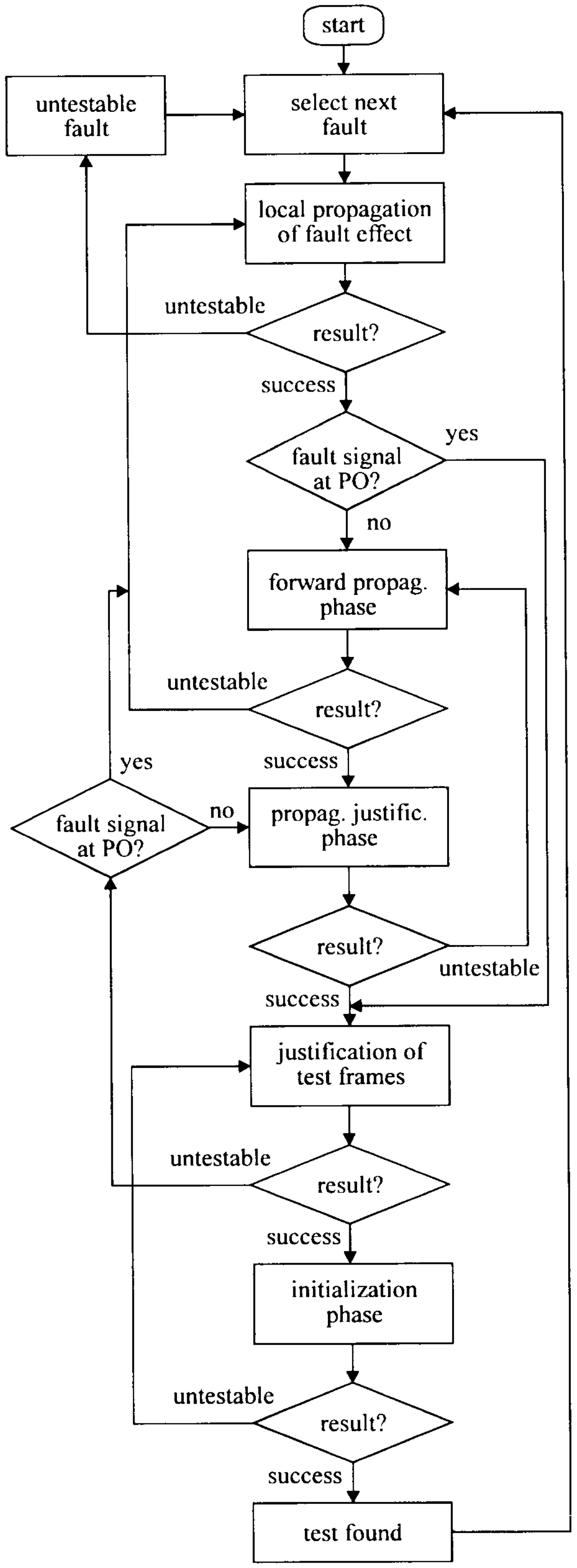

Figure 4: The extended FOGBUSTER algorithm

called. Since this process is performed in backward direction, attaching all of the time frames used during propagation, finally the fast clock time frame is reached. Then the local test generation is called for performing the propagation justification task for the fast clock time frame resulting into an init state to be synchronized.

The synchronization is again performed by SEMILET. For the synchronization a slow clock is used and thus the faulty machine does not differ from the good machine.

Note that the propagation justification and the synchronization phases both are performed by using reverse time processing while the propagation phase uses forward time processing.

The complete FOGBUSTER-algorithm is illustrated in Figure 4.

5: Fault simulation

After test pattern generation for a fault has been successful fault simulation is performed. In our approach this is divided into three phases:

1. Simulation of the good machine for all time frames of the initialization and for the fast clock frame. X-values left by the test generation are set at random to 0 or 1.
2. "Stuck-at fault simulation" of the propagation phase for all PPOs where possibly fault effects can occur at the end of the fast time frame, i.e. for all PPOs that have non-steady values.
3. Delay fault simulation of the fast time frame by critical path tracing for all POs, and for all PPOs that were observable at a PO in the propagation phase.

The first and the second part of the fault simulation are performed by FAUSIM, the fault simulator integrated in SEMILET, the third part is performed by TDsim, the fault simulator integrated in TDgen.

The first part of the fault simulation approach is only simulation since no fault can be detected in this part. In the second part a D or Dbar value is injected at each PPO that is not steady one ore zero. Then FAUSIM performs global fault simulation by handling the fault effect like a stuck-at fault that occurs only at the observation point (PPO) in the fast clock time frame. All later time frames don't consist of this fault, i.e. the good and the faulty don't differ at the observation point. This is done because only slow clocks are applied for the time frames handled by the fault simulator FAUSIM.

Finally TDsim performs a fault simulation for the fast time frame, by critical path tracing (CPT) for delay faults. This CPT is started at all POs and at all PPOs that FAUSIM found to be observable at a PO in the propagation phase. Faults detectable through a PPO are however only marked as detected if the fault effect cannot, as a side effect, invalidate the state that was required to propagate the fault to a PO in the propagation phase. These invalidations of detectability are detected by a separate CPT process, that is started from all PPOs with non steady values, or from the PPOs that were required to have fixed values to justify the propagation phase for faults detectable through the PPO used during test generation.

circuit	tested	untstbl	aborted	# pat	time [s]
s27	39	11	0	40	<1
s208	112	242	16	163	90
s298	164	260	110	1148	452
s344	313	199	100	494	403
s349	312	211	101	500	394
s386	332	335	77	390	80
s420	124	584	32	166	169
s641	807	136	211	560	310
s713	427	395	432	292	795
s838	113	1277	84	152	522
s1196	2114	69	13	1533	243
s1238	2181	136	13	1524	301

Table 3: Benchmark results

6: Experimental results

The combination of TDgen and SEMILET has been used to generate robust tests for delay faults in a number of the ISCAS89 sequential benchmark [21] circuits. Each line in each circuit was to be tested for a StR as well as a StF fault. As fault simulation is performed after each successful generation of a test, faults that were additionally tested by the generated patterns were not explicitly targeted by the test pattern generator. Table 3 shows the results of these experiments. For each circuit the total number of tested faults (second column), the number of untestable faults (third column) and the number of faults where test pattern generation was abandoned (fourth column) is shown. Test pattern generation was aborted after either 100 backtracks for the local test pattern generator, or 100 backtracks for the sequential test pattern generator. The fifth and sixth column show the total number of patterns generated and the total time needed for test pattern generation, on a Sun Sparc 10 Station, in seconds. The number of patterns generated as shown in the fifth column includes the patterns needed for initialization and propagation.

It is remarkable that for some circuits the number of untestable faults is quite high. Although some of these faults are combinationally redundant, a large part of these faults is only sequentially untestable, i.e. a test for the delay fault in the combinational part can be generated, but the fault effect cannot be propagated to a PO in the propagation phase, or the pattern cannot be initialised. A lot of these sequential untestables are due to the robustness criterion, that does not allow to specify a value for a PPO that changes it's value in between the initial and final time frame of the local test pattern generation: because of the fast clocking, the stabilization of the final value cannot be guaranteed. Only the values of the PPOs that have an equal initial and final value, without a hazard, can be specified by TDgen to SEMILET. The final value of PPOs that show a transition or a hazard cannot be specified robustly (without any chance to test invalidation). These values are handed over by TDgen as a special kind of don't care, that is unjustifiable: SEMILET must assume a fixed, but unknown value is present. Nevertheless it has to propagate the fault effect without this knowledge. This is a hard task if many of these values are present, resulting in a high number of faults that is untestable by this logic.

Future work will include the extension of the robust local test pattern generation to an approach in which the arrival and stabilization times of all signals are calculated, allowing a more precise indication of signal values at certain times. This will make the task of propagation of the fault effect easier, thereby making robustly untestable faults testable.

7: Conclusions

This paper presents an approach to robust gate delay fault ATPG for circuits without a scan path. It relies on coupling TDgen, a combinational test generator for gate delay faults with SEMILET, a sequential test generator using the FOGBUSTER-algorithm. Experimental results on benchmark circuits show that the number of untestable faults due to a strong robust delay fault model is large. This number is expected to be significantly decreased by using a non-robust fault model.

8: Acknowledgements

This work was supported by the European Community under the ESPRIT Project ATSEC (6575).

9: References

[1] G.L. Smith, "Model for Delay Faults Based upon Paths", *Proc. Int Test Conf. 1985*, pp. 342-349.

[2] C.J. Lin and S.M. Reddy, "On Delay Fault Testing in Logic Circuits", *Proc. Int. Conf. on CAD, 1986*, pp. 148-151.

[3] V.S. Iyengar, B.K. Rosen and I. Spillinger, "Delay Test Generation 1 -Concepts and Coverage Metrics", *Proc. Int. Test Conf. 1988*, pp. 857-866.

[4] V.S. Iyengar, B.K. Rosen and I. Spillinger, "Delay Test Generation 2 -- Algebra and Algorithms", *Proc. Int. Test Conf. 1988*, pp. 867-876.

[5] W.W. Mao and M.D. Ciletti, "A Variable Observation Time Method for Testing Delay Faults", *Proc. ACM/IEEE Design Automation Conf. 1990*, pp. 728-731.

[6] E.S. Park, B. Underwood, T.W. Williams and M.R. Mercer, "Delay Testing Quality in Timing Optimized Designs", *Proc. Int. Test Conf. 1991*, pp. 897-905.

[7] E.S. Park, and M.R. Mercer, "An Efficient Delay Test Generation System for Combinational Logic Circuits", *IEEE Trans. CAD*, Vol. 11, No. 7, July 1992, pp. 926-938.

[8] G. van Brakel, Y. Xing and H.G. Kerkhoff, "Efficient Test Pattern Generation for Delay Faults", *Proceedings IEEE/ProRISC Workshop on Circuits, Systems and Signal Processing*, Houthalen, Belgium, 1992, pp. 265-270.

[9] U. Mahlstedt, "DELTEST: Deterministic Test Generation for Gate Delay Faults", *Proc. Int. Test Conf. 1993*, pp. 972-980.

[10] C.T. Glover and M.R. Mercer, "A Method of Delay Fault Test Generation", *Proc. 25th IEEE DAC*, 1988, pp. 90-95.

[11] Y.K. Malaiya and R. Narayanaswamy, "Testing for Timing Faults in Synchronous Sequential Integrated Circuits", *Proc. Int. Test Conf. 1983, pp.560-571.*

[12] K.T. Cheng, S. Devadas and K. Keutzer, "A Partial Enhanced-Scan Approach to Robust Delay-Fault Test Generation for Sequential Circuits", *Proc. Int. Test Conf. 1991*, pp. 403-410.

[13] B. Koenemann et al, "Delay Test: The Next Frontier for LSSD Test Systems", *Proc. Int. Test Conf. 1992*, pp. 578-587.

[14] W. Cheng, "The BACK-Algorithm for Sequential Test Generation", *Proc. ICCD '88*, pp. 66-69

[15] W. Cheng, T. J. Chakraborty, "GENTEST: An Automatic Test-Generation System for Sequential Circuits", *IEEE Computer '89*, pp. 43-49

[16] M. H. Schulz, E. Auth, "Essential: An Efficient Self-Learning Test Pattern Generation Algorithm for Sequential Circuits", *Proc. Int. Test Conf. 1989*, pp. 28-37

[17] N. Gouders, R. Kaibel, "Advanced Techniques for Sequential Test Generation", *Proc. 2.nd European Test Conference*, Munich, 1991

[18] U. Gläser "Mixed Level Test Generation for Synchronous Sequential Circuits Using the FOGBUSTER Algorithm", *Arbeitspapiere der GMD 869.*

[19] S. Devadas, "Delay Test Generation for Synchronous Sequential Circuits", *Proc. Int. Test Conf. 1989*, pp. 144-152.

[20] J. Rajski and H. Cox, "A Method to Calculate Necessary Assignments in Algorithmic Test Pattern Generation", *Proc. Int. Test Conf. 1990*, pp. 25-34.

[21] F. Brglez, F. Bryant, D. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits", *Proc. 1989 Int. Symp. Circ. and Systems*