http://dx.doi.org/10.5755/j01.eee.21.3.5774 ELEKTRONIKA IR ELEKTROTECHNIKA, ISSN 1392-1215, VOL. 21, NO. 3, 2015

Acceleration of Fault Simulation based on a Separate List of Faults for Each Test Pattern

Rimantas Seinauskas[1], Ramunas Cvirka[1], Greta Rudzioniene[1]
[1]Department of Software Engineering, Kaunas University of Technology,
Studentu St. 50–404a, LT-51368 Kaunas, Lithuania
rimantas.seinauskas@ktu.lt

[1]***Abstract*—A new fault simulation procedure is suggested. The procedure provides fault detection of individual test patterns at the beginning. In this way, most faults are detected quickly. The remaining faults are analysed by conventional means with a sequence of test patterns. Creation of individual fault lists of test patterns allows speeding up the fault simulation. The fault simulation acceleration increases with circuit size and test coverage.**

***Index Terms*—Integrated circuit testing, failure analysis, fault simulation, model checking.**

I. Introduction

Functional test generation requires an assessment of test quality. For this purpose, fault simulation is used which takes a lot of time. Therefore, accelerating fault simulation is particularly relevant. The fault simulation acceleration problem has long been considered. The fault simulation task is very suitable for parallelization of computations. Improvement of fault simulation methods had a significant impact as well. This article proposes to make a fault list for each pattern of sequential circuits. A pattern consists of a sequence of input stimuli.

In general, the first stimulus of the test pattern uses the indefinite circuit state. The fault simulation object is a series of test patterns. The initial circuit state may establish a test pattern, which has been lodged before. In this case the test pattern can detect a fault which does not occur when the initial state is uncertain. Detection of these faults depends on the order of test patterns. We will call such faults as detected on the sequence of pattern. Detected faults on patterns are independent of the order of test patterns and are found in all of the initial circuit states. Functional test generation, minimization of test pattern sequence should be based only on faults detected on patterns. Evaluation of the final test could rely on the detection of faults on the sequence of patterns. Most of the tools of fault simulation determine faults detected on the sequence of patterns. Typically, test patterns are generated so that they detect faults in the circuit regardless of the initial state. Combining test patterns in a sequence, if successful, allows detecting additional faults that cannot be detected by an individual test pattern. It is difficult to tell how many faults can be detected in such a case in advance. It depends on the degree of fault coverage and luck. Therefore, it is not a reliable way to increase fault coverage.

Manuscript received November 23, 2013; accepted October 14, 2014.

Usually fault simulation inspects whether all undetected faults are detected by a test pattern. The list of the inspected faults is very large for complex circuits. Only a small percentage of inspected faults is confirmed as detected. Clearly unlikely faults can be rejected in advance. Pattern simulation results can be used to reject unlikely faults. The use of these ideas is described in the article.

II. Related Work

The main fault simulation studies were carried out mainly two decades ago and are widely described in many books and articles, but we will refer to only one of them [1]. The existing techniques for speeding up fault simulation provide algorithmic enhancements [2] and development of special-purpose hardware for fault simulation [3]. Fault injection approach, which is based on a co-operation between a simulator and an emulator, is presented in [4]. The number of faults that need to be simulated can be decreased by exploiting fault equivalence and fault dominance between a pair of faults [5]. Fault collapsing is used to reduce fault simulation time. It is the practice in which faults detected by a pattern are deleted from the fault list prior to the simulation of any subsequent pattern. Fault dropping decreases the complexity of fault simulation, but cannot be used for all fault simulation algorithms [6].

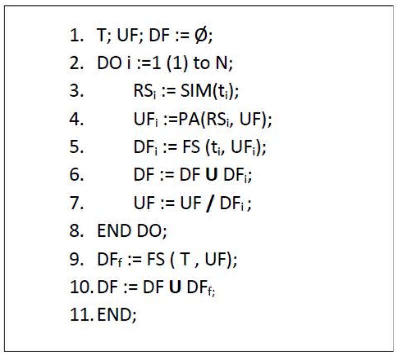
Fault simulation algorithms can be divided into serial, parallel, deductive and concurrent [7]. Serial algorithms simulate fault-free and faulty circuits and compare responses. Such algorithms are easy to implement; need only a true-value simulator, most faults, including analog faults, can be simulated, but they use lots of repeated computation. The parallel fault simulation takes advantage of multi-bit representation of data and availability of bitwise operations [8]. With each pass of the simulation, the fault-free circuit as well as machine word length faulty versions is simulated in parallel for a given pattern, but fault dropping cannot be used. Deductive fault simulation is a one-pass simulation, utilizes a dynamic data structures and 3-valued logic [9]. Computation rules are difficult to derive for complex gates and gate delays are difficult to use. Concurrent fault simulation is based on the observation that most values in most of the faulty circuits match the corresponding values in the good circuit [10]. Information about a fault will be entered in the fault list if at least one input or output of the gate is different from that implied at the corresponding line in the fault free version of the circuit. The fault is removed from the fault list if the corresponding input/output values are identical to that of the fault-free circuit.

Hardware fault simulation methodology and tools, using partial reconfiguration, is suitable for efficient fault modelling and simulation in FPGAs [3]. FPGA-based hardware fault simulation using partial reconfiguration is rewarding, as an alternative to software fault simulation, reducing fault simulation costs. Method of increasing the speed-up ratio of fault simulation in parallel test generation is presented in [11]. The method is based on fault partitioning.

Dynamic fault grouping based on fault activity is used in both HOPE [12] and the PROOFS [13] systems. Faults are grouped so as to initially detect more faults. Fault simulation time is reduced to the principle discard the detected faults. A Fault list is formed for all patterns. In this article, we'll offer a new fault grouping for each pattern, and thus reduce the fault simulation time. We did not find any articles describing such an approach. The list of faults for patterns opens up new opportunities to reduce fault simulation time for functional test generation [14].

III. FAULT SIMULATION WITH A SEPARATE LIST OF FAULTS FOR THE TEST PATTERN

Test T consists of a sequence of test patterns, where T = <t_1, t2, …, ti, …, tN>. In turn, a test pattern for sequential circuits is a sequence of stimuli. The input data of the fault simulation tool is a test T and the list of undetected faults UF. Fault simulation program FS determines which faults of the list UF are detected by test T and write them to the list of DF, it is DF = FS (T, UF). The proposed fault simulation acceleration procedure FSAP is shown in the Fig. 1.



```
1.  T; UF; DF := Ø;
2.  DO i :=1 (1) to N;
3.        $RS_i$ := SIM($t_i$);
4.        $UF_i$ :=PA($RS_i$, UF);
5.        $DF_i$ := FS ($t_i$, $UF_i$);
6.        DF := DF U $DF_i$;
7.        UF := UF / $DF_i$ ;
8.  END DO;
9.  $DF_f$ := FS ( T , UF);
10. DF := DF U $DF_f$;
11. END;
```

Fig. 1. Fault simulation acceleration procedure FSAP.

The first line of the procedure shows that the test T and UF set of undetected faults are given, and in the beginning of the procedure set DF of detected faults is empty. The cycle, which analyses all test patterns include lines from 2 to 8. Simulation of test pattern t_i is carried out in the third line of procedure and the results are denoted as RS_i. UF_i set of the most likely faults that can be detected by test pattern t_i is calculated in the procedure PA of the fourth line. The calculation is based on the simulation results RS_i of test pattern t_i. Also set UF contains still undetected faults to be assessed. Conventional fault simulation of a test pattern t_i with a set of expected faults UF_i is done in the fifth line. Calculated faults of the set DF_i are added to the set DF (line 6), and are discarded from the set UF (line 7).

The sequence of test patterns T can detect more faults than the one found in the analysis of individual test patterns. Therefore the remaining undetected faults should be analysed in the usual way, in order to determine which faults can be detected by using a sequence of test patterns. There is yet another reason. Generally, not all faults which are detected on test pattern t_i can be included in the set UF_i (line 4). Final fault test simulation of whole T and of the remaining undetected faults is shown in line nine. Faults that have been detected during the final fault simulation are added to the set DT of detected faults (line 10).

The size of the set UF of undetected faults which is left for final fault simulation determines the possibilities to accelerate fault simulation. It is obvious that the more faults are detected by test T, the higher the chance to speed up the fault simulation. Also, the smaller and more accurate set of UF_i is formed (line 4), the more fault simulation is sped up. Fault simulation time is almost linearly proportional to the amount of test patterns, when the amount of faults is fixed. Also, fault simulation time is almost linearly proportional to the amount of faults, when amount of test patterns is fixed. UF_i set of individual test patterns is less than the total fault set of UF. Let's say that on average, UF_i contains x % of the set UF. In this case, the individual test pattern fault simulation will take about x % of the time compared to the case where all the faults are analysed. Total fault simulation time can be shortened only when the final fault simulation will analyze less than (100 - x) % of fault models. Therefore the test T fault coverage must be no less than x %. This limitation becomes irrelevant if we can assure that the set of UF_i has all the faults detectable by the test pattern. However, the final fault simulation is still needed due to the fact that the sequence of test patterns detects more faults than as separate test patterns.

IV. EXPERIMENTS

In general, all the test patterns are constructed for the detection of faults, when the initial circuit state is uncertain. Side effects occur after the aggregation of test patterns in a sequence and test pattern sequence can detect those faults which are not detected by test patterns taken separately. The scope of the additionally detected faults depends on a variety of factors, and therefore the side effects are not sufficiently explored. We also will not make such studies, but because of a common perception we take just one example. First, we look at how many more faults a test pattern sequence detects over the individual test patterns, when the initial state is uncertain. For this purpose, we analysed the CPU of or_1200 processor. Tetramax tool automatically generated 1705 test patterns for the CPU. Test Generator showed that collapsed 49116 transition faults were detected. Fault simulation pointed out that the sequence of test patterns detects 47957 collapsed transition faults. Fault simulation of individual test patterns showed that they detect 41547 collapsed transition faults. Full set of fault UF was used in this case.

TABLE I. THE DIFFERENCE DETECTED FAULTS FOR TEST PATTERNS FROM 1 TO 900.

Patterns from	1	101	201	301	401	501	601	701	801
Patterns to	100	200	300	400	500	600	700	800	900
Patterns sequence	21583	6231	3461	1177	4815	3053	1080	855	1915
Separate patterns	13817	4528	3262	1731	3181	2969	1483	1269	2026
Difference	-7766	-1703	-199	554	-1634	-84	403	414	111

TABLE II. THE DIFFERENCE DETECTED FAULTS FOR TEST PATTERNS FROM 900 TO 1705.

Patterns from	901	1001	1101	1201	1301	1401	1501	1601	Total
Patterns to	1000	1100	1200	1300	1400	1500	1600	1705	
Patterns sequence	874	869	131	1417	198	304	318	188	47957
Separate patterns	1670	357	880	1501	748	644	617	352	41547
Difference	796	512	749	84	550	340	299	164	-6410

The difference between the detected faults sets have been analysed in more detail. The quantities of detected faults have been calculated for each one hundred of test patterns, when the fault simulation was carried out for all test sequences and test patterns individually. The results are shown in Table I and Table II.

The third and fourth rows show the newly detected fault quantities. The fifth row of the tables shows the difference between the quantity of faults detected by the individual simulated test patterns and by simulating the entire sequence. Fault simulation of a homogeneous sequence of the first one hundred test patterns detected 7766 more faults compared to the individual test pattern fault simulation. This is very surprising. Initially, the difference is mainly negative and by increasing the amount of test patterns becomes positive. Total maximum negative difference (-10382) was obtained after analysis of 600 test patterns. 600 test patterns detect about 40 % of the CPU transition faults. Further the difference decreases, and the difference should be negligible when approaching test coverage of 100 %.

Final fault simulation can also use not all of the remaining undetected faults, but only faults which can be verified by a given test. Determining of faults undetectable by test can be performed as well as by the individual test pattern. Now we will discuss most likely detected or undetected faults with regards to a given test.

Determining faults that are not detectable by the given test may rely on such an assumption. The fault of a gate, which is not detectable on output of the logic gate by a given test, is not detectable by the test as a whole on the output of the circuit. The values of gate outputs are set during the simulation. Faults detected in gate output can be set with the simulation. These faults can be detected by the test. Faults that are not detected on output of the logic gate cannot be dealt with on the basis of the assumption. Determining faults detectable on output of the logic gate requires the addition of a simulation program. Another way is an analysis of the simulation results. Faults that are detected on output of the logic gate are determined by the examination of simulation results. Simplified rules can be used to speed up calculations. We will discuss the potential impact of such rules.

The rules that define faults found in output of the logic gate depend on the fault type. It is easier to establish such rules for transition faults, because they depend on changes in values. More complex rules can evaluate not only the fault detection on output of gate but also memorizing failure on triggers. Such rules are heuristic and appropriate in the setting of only the most likely fault detection. Rules should not require a lot of calculations, and should choose only faults to be detected with high probability for normal fault simulation. The rules are the main factor that determines acceleration of fault simulation. Rules can be very different and their benefits may be proven only in an experimental study. This is not the purpose of this paper and is the object of further study.

The gate transition fault is not detected if the signal that is associated with the transition fault during test execution does not gain values zero and one. We will use this simple rule in an experimental study to assess the fault simulation acceleration using the proposed procedure.

Let us first examine how the formation of the list of faults of individual test patterns can speed up the fault simulation. Fault lists were formed based on the simulation results and the above rule. Ten first CPU circuit test patterns were analysed. Results are presented in Table III.

The second line provides fault simulation time in seconds for individual test patterns when the complete fault list was used. The third line provides fault simulation time in seconds for individual test patterns when the list of faults was calculated on the basis of simulation results. The last row shows the acceleration times. The last column represents average acceleration.

We can see that the acceleration of individual test patterns fault simulation is significant for the given circuit. Ten test patterns of or_1200 processor have been analysed to determine the change of acceleration trend when increasing the size of circuits. Results are presented in Table IV. We can see that the average acceleration increased. We hope that with the increasing size of the circuit, fault simulation acceleration also increases when their fault lists are formed for individual test patterns. This trend is encouraging.

In general, the list of faults can be formed not only for the individual test pattern, but for several test patterns. The results of this study on the CPU circuit are presented in Table V. Results are presented in groups of ten test patterns. We see that the acceleration of fault simulation decreased when the fault lists are formed for groups of test patterns. However, one can expect that in this case more faults will be detected than when examining only the individual test pattern.

In general, the best way to achieve maximum test coverage is the use of test patterns, each of which starts from the undefined state.

TABLE III. COMPARISON OF FAULT SIMULATION DURATION OF THE CPU CIRCUIT

Patterns	1	2	3	4	5	6	7	8	9	10	Aver.
Full list	13.8	10.4	10.9	12.8	11.1	13.5	10.9	10.7	11.1	11.4	129.4
Formed list	2.4	1.8	1.7	1.9	2.1	2.4	2.0	2.3	2.7	2.6	21.9
Times faster	5.8	5.8	6.4	6.7	5.3	5.6	4.7	4.7	4.1	4.4	5.9

TABLE IV. COMPARISON OF FAULT SIMULATION DURATION OF THE OR_1200 PROCESSOR CIRCUIT

Patterns	1	2	3	4	5	6	7	8	9	10	Aver.
Full list	9818	10037	12496	12418	12747	9937	9949	9913	12411	10008	109734
Formed list	475	549	691	756	715	546	459	475	502	723	5891
Times faster	20.6	18.3	18.1	16.4	17.8	18.2	21.7	20.9	24.7	13.8	18.6

TABLE V. COMPARISON OF FAULT SIMULATION DURATION BY USING SEQUENCES OF TEST PATTERN

Patterns	10	20	30	40	50	60	70	80	90	100	Aver.
Full list	91	89	84	76	74	79	80	70	57	63	763
Formed list	33	35	32	31	31	35	35	30	25	30	317
Times faster	2.8	2.5	2.6	2.5	2.4	2.6	2.3	2.3	2.3	2.1	2.4

Exploitation of side effects based on the fact that the sequence of test patterns detects additional faults is appropriate only when the separate test patterns cannot increase the test coverage. Also minimizing test length is difficult because the emission of test pattern, which does not detect new faults in the sequence, can change the fault coverage of whole patterns sequence.

List of faults can be formed according to the simulation results of all test pattern. Is it possible to guarantee that the faults are undetectable by the given test, if not entered into formed list? It depends on the rules for forming the fault list. We conducted an experiment using a rule; the transition fault can be detected by the given test if the electrical circuit of the fault had signal values zero and one.

We made the list of faults under the above rule for CPU circuit and for the 900 test pattern. A list of 92706 possible detectable faults was formed. CPU circuit has 126,986 faults in total. Fault simulation has detected the same amount of 44194 faults on both lists. Fault simulation, with a partial list took 40 percent less time in comparison with fault simulation with a full list. Naturally, the experiment cannot prove that the faults that were omitted under the rule really are undetectable. However, there is hope that the rule allows the selection of the majority of the faults that are potentially detectable.

Fault simulation acceleration based on the proposed procedure is based on rules that generate a list of faults of individual test patterns. These rules also determine whether the final faults simulation is required. In the future, such rules should be investigated in more detail.

V. CONCLUSIONS

Creation of separate lists of faults for test patterns reduces the fault simulation time. Fault lists of test patterns are drawn on the basis of simulation results of test patterns. Fault simulation time is proportional to the size of fault lists of test patterns. Fault list size determines the rules by which they are formed based on the simulation results of test patterns. Final fault simulation with the remaining undetected faults is necessary if the rules do not guarantee that all detected failures were included in the lists. Creation of individual fault lists is based on the assumption that the faults which are undetectable on the logic gate outputs are undetectable on circuit outputs as well. The sequence of test patterns can detect faults that are undetectable by analysing individual test patterns. This side effect occurs mostly when the fault coverage of the test is low. Influence of side effects decreases with increasing coverage of faults. The fault simulation acceleration increases with the size of circuits.

REFERENCES

[1] L.-T Wang, Y.-W Chang, K.-T Cheng, *Electronic design automation: synthesis, verification, and test*. Morgan Kaufmann, 2009.

[2] L. Berrojo, *et al*, “New techniques for speeding-up fault-injection campaigns”, in *Proc. IEEE Design, Automation and Test in Europe Conf. and Exhibition*, 2002, pp. 847–852. [Online]. Available: http://dx.doi.org/10.1109/date.2002.998398

[3] A. Parreira, J. P. Teixeira, M. Santos, “A novel approach to fpga-based hardware fault modeling and simulation”, *Design and Diagnostics of Electronic Circuits and Syst. Workshop*, 2003, pp. 17–24.

[4] A. Ejlali, *et al*, “A hybrid fault injection approach based on simulation and emulation co-operation”, *43rd Annual IEEE/IFIP Int. Conf. Dependable Systems and Networks (DSN)*, 2003, pp. 479–479. [Online]. Available: http://dx.doi.org/10.1109/dsn.2003.1209958

[5] V. Agrawal, A. V. Prasad, M. V. Atre, “Fault collapsing via functional dominance”, *IEEE Int. Test Conf. (ITC)*, 2003, pp. 274–274. [Online]. Available: http://dx.doi.org/10.1109/test.2003.1270849

[6] X. Lin, *et al*., “Timing-aware ATPG for high quality at-speed testing of small delay defects”, *IEEE 15th Asian. (ATS 2006)*, 2006, pp. 139–146.

[7] M. Ben-ari, *Principles of concurrent and distributed programming*. Pearson Education, 2006.

[8] R. M. Fujimoto, *Parallel and distributed simulation systems*. New York: Wiley, 2000.

[9] A. Bosio, G. Di Natale, “LIFTING: A flexible open-source fault simulator”, *17th. IEEE Asian Test Symposium, (ATS 2008)*, 2008, pp. 35–40. [Online]. Available: http://dx.doi.org/10.1109/ats.2008.17

[10] J. Hou, A. Chatterjee, “Concurrent transient fault simulation for analog circuits”, in *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 2003, vol. 22, no. 10, pp. 1385–1398. [Online]. Available: http://dx.doi.org/10.1109/TCAD.2003.818129

[11] S. Venkataraman, S. B. Drummonds, “POIROT: A logic fault diagnosis tool and its applications”, in *Proc. Int. IEEE* Test *Conf.*, 2000, pp. 253–262. [Online]. Available: http://dx.doi.org/10.1109/test.2000.894213

[12] H. K. Lee, D. S. Ha, “HOPE: An efficient parallel fault simulator for synchronous sequential circuits”, in *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 1996, vol. 15, no. 9, pp. 1048–1058.

[13] C. R. Graham, E. M. Rudnic, J. H. Patel, “Dynamic fault grouping for PROOFS: A win for large sequential circuits”, in *IEEE Proc. of Tenth Int. Conf. VLSI Design*, 1997, pp. 542–544. [Online]. Available: http://dx.doi.org/10.1109/icvd.1997.568204

[14] E. Bareisa, *et al*, “On delay test generation for non-scan sequential circuits at functional level”, *Elektronika Ir Elektrotechnika*, 2011, no. 3, pp. 67–70. [Online]. Available: http://dx.doi.org/10.5755/j01.eee.109.3.173