Science Journal of Circuits, Systems and Signal Processing
2016; 5(2): 8-18
http://www.sciencepublishinggroup.com/j/cssp
doi: 10.11648/j.cssp.20160502.11
ISSN: 2326-9065 (Print); ISSN: 2326-9073 (Online)

Design Flow Allowing the Effective Use of Non-scan and Scan-Based Tests

Rimantas Seinauskas

Software Department, Informatic Faculty, Kaunas University of Technology, Kaunas, Lithuania

Email address:
rimantas.seinauskas@ktu.lt

To cite this article:
Rimantas Seinauskas. Design Flow Allowing the Effective Use of Non-scan and Scan-Based Tests. *Science Journal of Circuits, Systems and Signal Processing*. Vol. 5, No. 2, 2016, pp. 8-18. doi: 10.11648/j.cssp.20160502.11

Received: August 15, 2016; **Accepted**: August 23, 2016; **Published**: September 9, 2016

Abstract: At speed delay testing is important for embedded systems. Attempts to solve the problems of delay testing only with non-scan or scan-based tests are unsuccessful. There is no need to oppose these tests, but it is necessary to use both taking full advantage of their opportunities. Design flow and the ability to use non-scan and scan-based ATPG, functional test and fault simulation is presented. The goal is to detect as many faults with non-scan at-speed test. The remaining faults are detected with a scan-based test. As a result, there are less of undetected faults and the length of the scan-based test is reduced. The proposed approach provides more flexibility for test generation. Design flow forced the development of new methods for speeding up fault simulation and for more efficient generation of input patterns. Experimental results demonstrate the possibilities of approach.

Keywords: Non-scan Test, Scan-Based Test, Functional Test, Design Flow

1. Introduction

Circuit testing using only the inputs and outputs is the easiest way. The test generation problem is too complex to be resolved within a reasonable period of time for large circuits. Therefore, the additional inputs and outputs are added to change and monitor the internal circuit states. As a result, circuit size and testing time increases. Currently, large circuits are designed to facilitate testing. At speed testing of delay faults is particularly important for modern circuits. At speed testing detects defects that cannot be detected with a scan-based test. Scan-based test is usually voluminous and its implementation requires a lot of timing signals. Scan-based test can detect faults, which actually do not affect the functioning. In this case, there is a so-called problem of over-testing. Using together the non-scan and scan-based tests is possible. We are working with the assumption that at-sped test of transition faults detects more defects than the scan-based test of transition faults. Part of the faults can be tested with non-scan test and the other part of faults with the scan-based test. Non-scan test received only for part of faults also makes sense because it is possible to detect more defects through at speed testing and to shorten the scan-based test. Non-scan test is useful when using testing with partial scan as well.

Test generation methods have been developed for several decades. Commercial test generation packages contain circuit test generation tools without scan (full sequential). However, these tools can successfully generate tests only for those faults which do not require long test patterns (sequences). Large circuits, usually have only a small amount of faults detectable with short patterns. Therefore, the quality of the test is unsatisfactory and the test must be supplemented.

Fully sequential test generator test may be supplemented by adding a functional test. The functional test is usually generated by a higher level of abstraction and is usually long. It can detect the same faults as the test obtained using fully sequential generators. Therefore, the test patterns, which do not detect new faults, have to be discarded from the functional test. It can perform fault simulator.

Fault simulator can be used for the further addition of test quality. This article explores the various options for the addition of the test, using fault simulation. Scan-based test can detect the remaining faults that are not detected by the non-scan test.

The remaining article part is structured in such a way. Next Section 2 is dedicated to a brief overview of the most

important trends in test generation. Design flow, which allows flexibility in the use of non-scan and scan-based test generation approaches and gradual addition of the test are described in the third section. The fourth section explores test generation using state-of-the art non-scan ATPG. Fault simulation acceleration of functional test is discussed in the fifth section. The sixth section discusses the addition of similar test patterns to the final test. Experimental results with benchmarks are described in the seventh section. The conclusions of the article are presented in the last section.

2. Related Work

Scan-based testing achieves high fault coverage, but requires long test application times and substantial tester memory, in addition to the overhead in chip area and high test power. Non-scan test, on the other hand, suffers from low coverage, but can be applied at-speed. Experimental studies [1] have shown that some of the defects can be detected only by a functional at-speed test. Exclusion of at-speed test increases the risk that not fully tested chips will cause a system failure. Non-scan at-speed testing remains necessary despite all the effort [2] to improve the scan-based testing. At-speed tests are effective in relation to the length of the test, so aim to detect more faults with non-scan test is understandable.

The RAMs are tested using BIST, and are not covered by any scan test. Without the provision of the BIST tests a scan-based test would have substantial numbers of defective parts escaping. In this way, the non-scan and scan-based tests, as well as BIST should be used for testing of modern circuits

Test generation methods for non-scan circuits are developed for a long time [3, 4, 5, 6]. Design-for-Test methodology for non-scan at-speed testing is suggested in [7]. The circuit state set is expanded by use of inverse outputs of flip-flops, and grouping them by introducing an additional input enabling and facilitating the availability on the output. This provides better fault detection conditions. Design-for-Test methodology for non-scan testing at a functional level is suggested in [8]. Combining BIST and ATE are discussed in [9].

Functional test generation uses functional fault models and criteria based on the description of functioning on a higher level of abstraction [10]. Description can be expressed by algorithmic language before circuit synthesis.

Functional faults are associated with the text of the description [11]. Description of functioning at a higher level of abstraction allows faster processing of large circuits. However, detection of functional faults does not guarantee the detection of gate faults. Methods for generating functional tests allow the detection of faults that are not detected by test generation techniques at the gate level. Therefore, functional and gates-level test generation techniques are used in combination [12].

Test generation methods express conditions for faults detection and search for a solution that satisfies those conditions. For this purpose, it is necessary to tackle the system of constraints [13], as conditions may require conflicting values in the input. This poses a major problem of finding a solution. Methods of satisfying constraints are widely used in solving the problem of test generation. Computer science to solve the problem of satisfaction (often abbreviated as marked in capital letters - SAT) must determine whether the Boolean variables can be assigned so that the expression should result in a value equal to one.

Binary satisfaction is probably the most studied combinatorial optimization / search problem. Great efforts have been attempted to provide an efficient solution to practical problems. The SAT is one of the main problems in computer science with a theoretical and practical importance and has a very efficient practical realization [14]. SAT solvers are widely used for functional and gate-level test generation.

Search field increases considerably during the search of the test vector sequence for sequential circuits. Only relatively short sequences manage to calculate within a reasonable period of time when using deterministic methods. Therefore, functional test generation techniques are useful, which uses the principle of the selection [15]. Test patterns are generated randomly or according to the given rules. Test patterns that satisfy the quality criteria are selected as tests. In this case, the length of the test patterns is not a critical parameter. It is also possible to use complex quality criteria. The process efficiency depends on the rules used for test generation and quality criterion used.

Test quality criteria usually is based on simulation results of the test pattern and do not directly relate to the detection of gate faults. Therefore, functional test generation becomes inefficient when there are few undetected faults. In this situation the selection of test patterns may use fault simulation. Fault simulation is related to the long taking calculations. Calculation time reduction relates to fault list manipulation. The principles that are used for the generation of functional test can be applied using faults simulation [15]. Fault simulation requires greater resources than the analysis of functional faults. Therefore, faults simulation acceleration is relevant.

The main fault simulation studies carried out mainly through two decades ago and are widely described in many books and articles, but will refer to only one of them [16]. The existing techniques for speeding up fault simulation provide algorithmic enhancements and development of special-purpose hardware for fault simulation. The number of faults that need to be simulated can be decreased by exploiting fault equivalence and fault dominance between a pair of faults. Fault collapsing is used to reduce the 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.

Fault simulation algorithms can be divided into the serial, parallel, deductive and concurrent. Serial algorithms simulate fault-free and faulty circuits and comparing the responses. Such algorithms are easy to implement; need only a true-value simulator, most faults, including analog faults, can be

simulated, but use much repeated computation. The parallel fault simulation takes advantage of multi-bit representation of data and availability of bitwise operations. With each pass of 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 structure and 3-valued logic. Computation rules are difficult to derive for complex gates and gate delays are difficult to use. Concurrent fault simulation is based on a factor that most of the values in most of the faulty circuits agree with the corresponding values in the good circuit. Information about a fault will be entered in the fault list if the value implied 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. Measured coverage in the sample is used to estimate fault coverage in the entire circuit. It allows saving in computing resources, but limited data on detected and undetected faults is available.

Hardware fault simulation methodology and tools, using partial reconfiguration, suitable for efficient fault modeling and simulation in FPGAs [17]. FPGA-based hardware fault simulation using partial reconfiguration is rewarding, as an alternative to software fault simulation, constraining fault simulation costs. Usual clock frequencies make hardware fault simulation time two orders of magnitude less dependent on the number of patterns than software fault simulation. Method about how to increase the speedup ratio of fault simulation in parallel test generation is presented in [18]. The method is based on fault partitioning.

Dynamic fault grouping based on fault activity is used in both HOPE [19] and the PROOFS [20] systems. Faults are grouped so as to be initially detected more fault. Fault simulation time is reduced on the basis of the principle drop detected. Fault list is formed for all patterns. In this article, we'll offer up a new fault grouping for each pattern, and thus reduce the fault simulation time. The list of faults for patterns opens up new opportunities to reduce fault simulation time.

Functions for embedded systems can be implemented as hardware or as software. Generating integrated test requires disposing of a unified embedded system model. Binary FSM model unanimously represents the functioning of the hardware and software. Hardware and software test quality criteria can be used for the generation of functional integration test for embedded systems. Hardware test quality criteria are more stringent compared to software test quality criteria]].

3. Design Flow

Design flow, which allows flexibility in the use of non-scan and scan-based test generation approaches and gradual addition of the test is shown in Figure 1. The software prototype can be developed in parallel with the specification and can be used for verification of the specification. The circuit is synthesized with a full-scan register. Non-scan ATPG can generate a test that detects more faults on circuits with a scan and with additional inputs and outputs compared with the circuit without a scan. Functional test generation based on the software prototype may be carried out in parallel with the circuit synthesis. The functional test cannot use full-scan capabilities as based on the prototype, which was made before the circuit synthesis.

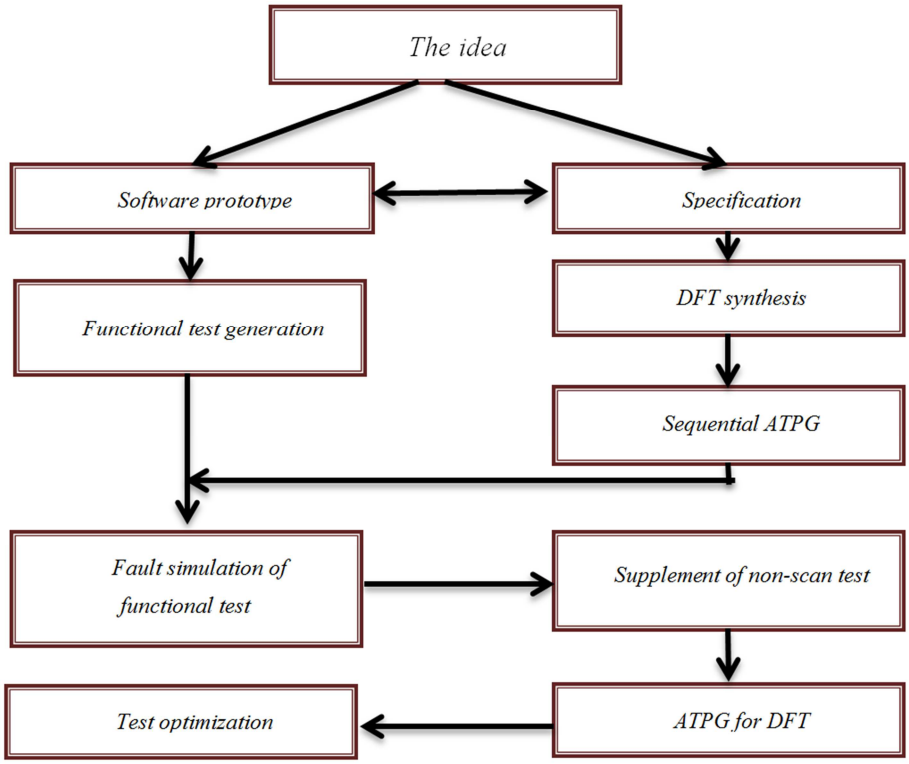

Fig. 1. *Design flow.*

Non-scan and scan-based tests will be generated. First non-scan test is generated for scan-based circuit. Full sequential ATPG generates a compact test and it is intended to take full advantage of the capabilities of the generator. Fault simulation shall select patterns, which detect new faults, from functional test. Further, the test is extended exploiting fault simulation capabilities. The remaining undetected faults are the object of the ATPG for scan circuits. Finally, non-scan and scan-based tests are optimized. Detection of more faults with non-scan test is always useful. This problem will be examined in this article. The proposed design flow is to be used in the case where very high requirements are raised for the coverage of faults, and computing time is not critical. Basic computing time costs are related to the generation of functional test that can be performed in the early design stages in parallel with the design steps. Quite a lot of time can be spent on this work, because it does not increase the time placing the product on the market. Design flow uses the principle of gradual addition of the test using different methods of replenishing.

4. Test Generation Using State-of-the Art Non-scan ATPG

DFT synthesis adds two additional inputs. Full sequential ATPG can use these two additional inputs. Test generation should terminate when the received test detects all faults. However, in practice it is difficult to reach. Therefore, test generation time should be limited. The test generation termination condition usually described by time, which is given on average for single fault detection before test generation.

Setting test generation termination conditions expressed in the average value is not trivial. The calculation time depends on a variety of limitations, which are referred for test generation. As an example, we will demonstrate the generation of tests for the ITC benchmark B14 using TetraMax program. Limitation of time for a finding of the test pattern is indicated in the program.

Table 1 shows the results of test generation with the full sequential generator of TetraMax. Time limits for single-fault test generation are shown in the first row. Test generation took several iterations gradually increasing allocated time limit. Time spent in a test generation iteration is shown in the second row. Amounts of detected faults for iterations are shown in the third row. Only undetected faults are examined during iterations. The fourth and fifth rows show the amount of test patterns (sequences) and the total amount of vectors. Amount of undetected faults after the iteration is shown in the penultimate row. The last line shows the calculated efficiency obtained by dividing the iteration time to the amount of detected faults.

Almost new faults are not detected during the fourth iteration, when limitation equal to four. This indicates that the ATPG possibilities are exhausted. In the last column, the results are shown when at the very beginning the time limit shall be equal to three. Summary results of all three iterations are shown in the penultimate column. In this case there is a fault decreases in comparison with the use of several iterations. The best quality of the test (number of detected faults) is obtained by generating a test in an iterative manner and gradually increasing the time for finding of test pattern for a single fault. It has been observed for other circuits as well. This small example demonstrates that the results of full sequential test generation significantly depend on limitations used and at the same time from the tester's skills. The same is true in case of the full scan ATPG use. This experiment shows that with the same tool quite different qualitative results can be obtained. The use of various limitation methods for computing time and the circuit scale is not the object of this article. We want only to say that the best-known in practice limitation methods used for ATPG test generation. This is important to show the influence of various test generation techniques to the final test quality.

It is appropriate to examine the obtained test patterns with a fault simulator, which accurately indicates faults that are detected by the generated sequence of patterns. In general, the best experience with the use of modern ATPG tools is fully used in further experiments.

Table 1. *Non-scan test generation for B14 benchmark.*

Time limit	1	2	3	4	Σ1, 2, 3	Ø-3
Generation time (sec.)	7522	11309	20191	4200	39022	16473
Detected faults	13789	2348	2859	1	18996	14929
Patterns	324	70	169	1	563	283
Total quantities of vectors	2142	480	1257	12	3879	1939
Not detected	12755	10407	7548	7547	7548	11615
Efficiency	0,55	4,81	7,06	4200	2,05	1,10

5. Acceleration of the Fault Simulation of the Functional Test

The functional test can be generated at the beginning of the design, when device software prototype is available. Functional test generation based on a functional fault model. The correlation between functional and gate faults models is not rigid. Therefore, some test patterns of functional test cannot detect new gate faults. Elimination of unnecessary test patterns is needed when the functional test is used as a complement to already existing test. Fault simulation can throw away unnecessary test patterns.

Elimination of test patterns requires that each test pattern to be analyzed independently from the others. Each test pattern to be simulated starting from the initial undefined state. The ability to detect some faults when the preceding pattern determines the state cannot be used during fault simulation.

Fault simulation time can be very high for large circuits and long functional tests. Therefore, the possibility of shortening the duration of fault simulation is important.

Currently, fault simulation methods and tools are fully developed. Fault simulation time mainly depends on the fault list size. Initially, the list of undetected faults is established and after the analysis of each test pattern it is adjusted. We suggest to create a list of faults detectable on each test pattern based on the simulation results. Each detected fault of test pattern to be detectable at the output of the logic gate. This assumption allows reducing the list of faults that are analyzed.

During the simulation logic gate value calculation can be easily linked to the determination of fault detectable on the output. This requires a modification of the logic simulation program. The same can be done with an additional program that analyzes the simulation values. Gate transition faults can be detected, if at least one of the input or output signal is changing. Based on this assumption software has been developed. The program forms a list of faults detectable on test patterns. This program reduces the duration of fault simulation. As far as we know such an approach has not yet been used.

```
1.  T; UF; DF := O;
2.  DO i :=1 (1) to N;
3.      RSi := SIM(ti);
4.      UFi :=PA(RSi, UF);
5.      DFi := FS (ti, UFi);
6.      IF (DFi ≠ O) THEN
7.          DF := DF U DFi;
8.          UF := UF / DFi ;
9.      ELSE T : = T / {ti}
10. END DO;
11. END;
```

Fig. 2. *Fault simulation acceleration procedure.*

Test T consists of a sequence of test pattern, where T = <t_1, t_2,…, t_i,…, t_N>. In turn, test pattern for sequential circuits is a sequence of input vectors. The input data of the fault simulation tool are 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 is shown in the Fig. 2.

The first line of the procedure shows that the test T and UF set of undetected faults are given, and at the beginning of the procedure set DF of detected faults is empty. The cycle, which analyzes all test patterns include lines from 2 to 10. 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 still undetected faults to be assessed. Faults are not included in the UF_i set if their gate inputs and outputs do not change during the simulations. This speeds up the fault simulation. Conventional fault simulation of a test pattern t_i with a set of expected faults UF_i is done in the fifth line. Test patterns, which do not detect new faults (line 6) are emitted from the test set T (line 9). Calculated faults of the set DF_i are added to the set DF (line 7), and are discarded from the set UF (line 8).

Let us first examine how the formation of the list of faults of individual test patterns can speed up the fault simulation. List of faults was formed on the basis of the assumption that the transition faults of gate cannot be detected if the signals of the gate do not change during logical modeling of a test pattern. For this purpose, the CPU of OR_1200 processor was analyzed. Tetramax tool automatically generated test patterns for the CPU. Ten first CPU circuit test patterns were analyzed. Results are presented in Table 2. The second line in seconds provides fault simulation times for individual test patterns when the complete faults list was used. The third line in seconds provides fault simulation times 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 an average of acceleration. We can see that the individual test patterns fault simulation acceleration is significant for a given circuit. Further ten test patterns of or_1200 processor (which includes CPU core) have been analyzed to determine the change of acceleration trend in increasing the size of circuits. Results are presented in Table 3. The average acceleration increased. It is the hope that with the increasing scope of circuit, fault simulation acceleration also increases when their fault lists are formed for individual test patterns. This trend is encouraging.

Table 2. *Comparison of fault simulation time 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	12.9
Formed list	2.4	1.8	1.7	1.9	2.1	2.4	2.0	2.3	2.7	2.6	2.19
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 3. *Comparison of fault simulation time 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	10973
Formed list	475	549	691	756	715	546	459	475	502	723	589
Times faster	20.6	18.3	18.1	16.4	17.8	18.2	21.7	20.9	24.7	13.8	18.6

Described fault simulation acceleration principle has two disadvantages. Fault list drawn up by the simulation results can not miss faults that can be detected with the present test pattern. List inaccuracy affects the final results. An acceleration efficiency decreases with decreasing undetected fault list. Acceleration is useless when there is a small undetected fault list. Acceleration tool should be used with caution. The fault simulation acceleration tool was used for the analysis of functional test and for the test addition.

6. The Addition of Non-scan Test Based on the Results of Fault Simulation

A functional test generation often uses random generation of test patterns and their selection according to the chosen criteria. Random generation allows the flexibility to choose the test pattern length. The selection criteria only indirectly reflect detectable faults. Therefore, functional test cannot achieve a high coverage of fault detection. The selection of test patterns based on the fault simulation can be meaningful as well. Fault simulation is an accurate criterion for the selection, but requires more computing resources. In this case the quantity of the analyzed generated test patterns will be less. However, a more accurate criterion can yield benefits. Straightforward policy use of fault simulation for test generation is not viable. A smart way of generating input sequences substantially changes the situation. Experiments confirmed that.

Setting the conditions for completing the generation, selection of the correct length of test patterns, setting the best distribution of ones and zeroes for random generation can be carried out on the same principles as during functional test generation. These principles are described in the article [15].

Evolutionary programming and genetic algorithms are often used for functional test generation. A method for generating mutations and fitness function are essential in this case. Fault simulation can be used as an accurate fitness function. However, fault simulation is receptive to to computer resources, and does not allow consider a lot of mutations.

Selection of test patterns based on the fault simulation creates preconditions for successful use of the principle of similar test pattern (mutation) generation, which summarizes the evolutionary programming and genetic algorithm. Each test pattern of T test detects any of circuit faults. The test pattern is similar to one given if only one or more input values are different. A similar test pattern can detect other faults. Similar test pattern of a given test may detect new faults. Similar test pattern is generated by changing some of the input values to the opposite. Inputs for replacement are selected in various ways. A test sequence that has all of the input values opposite in respect of a given sequence is 100 percent different and zero percent similar. A test sequence that differ only in one input value is the minimum differentiated and of the maximum similarity. The selected percentage influences the efficiency of generation.

CPU core of OR_1200 processor was selected for case experimentation. A situation was considered where a test has detected 53555 transition faults. Similar test patterns were generated for two test patterns. Maximum 300 similar test patterns were generated for different percentages of modifiable signal value. The results are given in Table 4.

Similar test patterns were generated for two test patterns, which were obtained at the end of test generation. One of them has detected 121 new transition faults, and the other just one. This was done to determine whether the amount of the detected faults is affecting the success of a similar test pattern generation. A similar test pattern generation is done by changing a fixed percentage of values to the opposite. The percentage of modified inputs are indicated in the first row. Table cells contain how many new faults are discovered after the generation of 100, 200 and 300 test patterns. In parentheses, it is indicated how many test patterns detected new faults. The column labeled "new" is intended for the results of pure random generation. It is presented to compare the results of a purely random and similar test sequence generation. The comparison clearly demonstrates that a random test sequence generation descends in respect of a similar test sequence generation.

Table 4. *Results of generating similar test patterns.*

Modifiable input percentage			**New**	**1%**	**2%**	**5%**	**10%**	**15%**	**20%**
Test patterns	**Detected new faults**	**Similar generated**							
		100	13 (2)	352 (23)	471 (27)	623 (30)	679 (24)	254 (14)	252(14)
1	121	200	41 (6)	445 (30)	578 (41)	829 (47)	905 (39)	347 (25)	298(22)
		300	50 (9)	450 (32)	614 (48)	891 (55)	979 (47)	380 (33)	396(28)
		100	7 (5)	165 (5)	191 (12)	250 (20)	261 (18)	378 (25)	130(19)
2	1	200	14 (8)	199 (7)	261 (20)	341 (31)	388 (32)	463 (35)	141(24)
		300	91 (9)	218 (12)	272 (24)	451 (40)	447 (43)	501 (43)	279(33)
Average		300	71(9)	334 (22)	443 (36)	671 (48)	713 (45)	441 (38)	338 (31)

Analysis of the experiment results shows that considerably better similar test patterns are generated from the test pattern that detects more new faults. Most of the new faults are detected during the first two hundred of the test patterns. The average number of detected faults (Table 4) by generating a test sequence with different percentage of modifiable inputs

is shown in Figure 3. The best similar test patterns are obtained when 10 percent of test values are modified for this circuit (Fig. 3.).

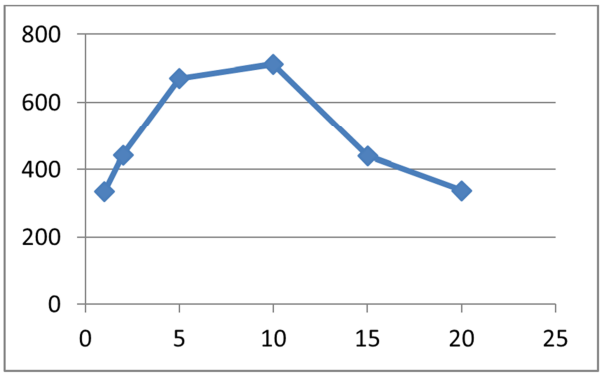

Fig. 3. *The average number of detected faults by generating with different percentage of modifiable inputs.*

The additional experimental study showed that the best modifiable inputs percentage depends on the number of previously detected faults. The best modifiable inputs percentage decreases with increasing quantities of previously detected faults.

The results reflect the regularities of one particular circuit. It is necessary to carry out experiments with lots of circuits, and with lots of test patterns to generalize regularity. However, generalized results cannot be accurate for a given circuit. Therefore, it may be appropriate to examine each individual circuit, in order to choose the best test generation strategy. Still, this approach is promising. We will use the results of such rapid study for generation of test patterns.

The assumption that it is appropriate to modify the less inputs, when remains little of undetected faults, allows us to create a general test generation method for cases in which there is little of undetected faults.

Minimal modification is the replacement of one binary value of the test sequence to the opposite. Minimal modifications often do not change the output values of the test sequence. It is therefore possible to use the assumption that it is unlikely that the test sequence with the same output values detect different faults. Therefore, modifications which do not change outputs of the test sequence, it is reasonable to be rejected. This significantly reduces the amount of re-calculation of fault simulation. Sample analysis showed that about 20 percent of the modified inputs of the test sequence replaces at least one output value.

These observations allowed us to create new minimal modifications (MM) approach. The approach consists of two parts. First all minimal modifications in test sequence are examined. Only those modifications which change the test sequence outputs are selected and analyzed with fault simulation.

The test sequence inputs that modification alters the outputs are written to a separate input set. Pairs of minimal modifications of such an input set are considered during the second phase. Pairs of minimal modifications that do not change the outputs of the test sequence are also rejected.

The new iterative procedure for test addition was formed on the basis of the observed trends. The procedure is presented in Fig. 4.

The procedure is used when test generation tools do not receive the required quality T test. Baseline data of the procedure are the lists DF, UF of detected and undetected faults, and test T received (line 1). T test is also written to the operating set of test patterns (TT, line 2). External loop DO1 (3-16 lines) may be automatic and not when the solution of ordinary iteration execution is taken after analysis of the results. TS set of test patterns, which detected only new faults, is selected from the set of available tests TT (line 4). Operating set TT is cleaned prior to the generation of patterns (line 5). Cycle DO2 (6-14 lines) deals with all the test patterns of the TS set. Cycle DO3 (7-13 lines) generates similar test patterns. Size SA determines the amount of generated test patterns.

```
1.  DF; UF;T;
2.  TT := T;
3.  DO1 it :=1 (1) to IT;
4.    TS : = SEL(TT);
5.    TT := Ø
6.    DO2 i :=1 (1) to END of TS;
7.      DO3 j :=1 (1) to SA;
8.          t_i := MM(t_i, j)
9.          DF_i := FS (t_i, UF);
10.         IF ( DF_i ≠ Ø) THEN
                TT:=TT U {t_i};
11.         DF := DF U DF_i;
12.         UF := UF / DF_i;
13.     END DO3;
14.   END DO2;
15.   T := T U TT;
16. END DO1;
```

Fig. 4. *The new iterative procedure for test addition.*

Procedure MM for minimal modifications always generates a new input sequence maximal similar to the test pattern t_i (line 8). Generated input sequence changes at least one output. The parameter j refers to generating consistency. Minimal modifications are generated first and then pairs of minimum modifications are generated. SA factor limits the amount of generated similar input patterns. Less similar input patterns are generated when the all minimum modifications and their couples are being used. Fault simulation estimates the faults detected by the input sequence generated (line 9). Actions of increased efficiency for fault simulation is described in Fig. 2, so there is no repeat. The test pattern t_i is added to the set TT, if it identifies at least one new fault (line 10). Rows 11 and 12 adjust the lists of the detected and non-detected faults. At the end of the iteration T test is extended (line 15).

Execution time of the procedure depends on the duration of fault simulation of a test pattern. The overall procedure's execution time is proportional to the multiplication product

of the selected test patterns (line 4) and the amount of similar test patterns generated (SA, line 7). Therefore, the duration of a single iteration of the procedure can be roughly calculated.

A careful analysis of the results obtained in the last iteration of the procedure enables to choose a reasonable amount of the generated similar test patterns and the amount of test patterns, which are selected to the set TS. Selection of the iteration parameters provides the flexibility and adaptability during the task solution. It is also possible to examine the influence of the distribution of ones and zeros of the tests. Additionally, it is also possible to construct tests manually and carry out generation of similar tests.

The table 5 shows how iterations converge by generating similar test patterns. Iterations started with a test of the 2142 test patterns that detect 53555 transition faults. Five test patterns were selected in the first iteration, and generated 300 similar tests, of which 135 tests detected 1470 faults. In the first iteration, the amount of the detected faults decreases rapidly, but the decline has slowed down later. Five test patterns that detect the majority of new faults are selected for generating similar test patterns. The question of how much it is appropriate to select test patterns for further generation still requires further investigation. Within ten iterations 550 test patterns were selected, which detected 4040 faults (last column). An iteration should end when new faults are not detected at all or the amount of newly detected faults is very low.

The proposed iterative procedure for the addition of the test most appropriate to use in the final phase of the generation when other generation methods have become ineffective. Similar input pattern generation is most successful when the received test already detects more than 95 percent of faults. This is very important when we need to achieve the highest possible level of fault detection. The proposed addition of the test procedure can be used not only for the final test generation phase. In this case, we need to select the best degree of similarity of generated input sequences of the procedure MM.

Table 5. *Iterations of similar test pattern generation.*

Iterations	0	1	2	3	4	5	6	7	8	9	10	Total
Selected	0	5	5	5	5	5	5	5	5	5	5	
Faults det.	53555	1470	633	441	240	180	244	207	282	204	143	4040
Patterns	2142	135	75	66	41	35	46	37	38	39	38	550

7. Experiments with Benchmarks

Design flow, which is shown in Fig. 1 provides that the synthesized circuit has automatically included DFT accessories. It can be full scan or accessories that increase circuit observability and controllability. This decision is very important because the tests are immediately generated for the modified circuit. Additional inputs of circuits allow detecting more faults with non-scan test. However, the test patterns which are obtained with a state-of-art ATPG are limited to a short vector sequence. Therefore, the following functional test can significantly increase the amount of the detected faults. The method for generating of functional test is described in [15].

The functional test is generated for a circuit without scan. The additional inputs for a scan must define the operating mode of the circuit. The functional test does not evaluate that some of the faults have detected by non-scan test. In addition, some test patterns cannot detect new faults, especially for a generalized selection criterion. Therefore, fault simulation enables to discard redundant test patterns.

Fault simulation accurately determines whether a test pattern detects faults still undetected. Test patterns can be generated at random and selected only those who find new faults. Drawing up lists of faults that can detect a particular test pattern (described in section 5) enables efficient selection of test patterns from a randomly generated. Random generation remains meaningful and after application of a functional test to detect yet undetected faults. In this way, addition of test coverage is done.

Another way to add to the test coverage based on the generation of similar test patterns as described in section 6. When all the possibilities of non-scan test generation are exhausted scan-based ATPG is used for the remaining faults. This allows not only increase the amount of faults found by the non-scan test, but also to reduce the amount of undetected faults and the amount of clock signals. This is illustrated by the experimental results with the benchmarks.

Non-scan and scan-based test optimization uses sorting of test patterns. Test patterns which detect more faults are transferred forward. This allows some of the test pattern throw out. Latest vectors of test patterns can be discarded if they do not entail new fault detection.

Biggest benchmarks B14, B15, B17, B20, B22 of ITC'99 set [22] and publicly available CPU of OR_1200 processor were selected for experiments. Test generation time, the quantity required clock signals and remaining undetected faults are shown in Table 6 for each test generation method. Data on test generation with fully sequential test generator Tetramax are shown in the third column. Tests were generated for the circuits with scan and the two additional inputs. Generation time constraints were chosen in order to get the best performance. The fourth column contains the data for fault simulation of functional test, that is, the duration of fault simulation (Time(s)), how much clocks required of the selected test patterns to detect new faults (Clocks), and the remaining undetected faults after the functional test (Not detec.). Functional test has been obtained on the basis of the methods described in [15]. Subsequent two columns show data after the selection of the test patterns generated at random and generated similar to the already selected as described in the fifth and sixth sections. Next TetraMax scan ATPG was used for the remaining undetected

faults. The data are presented in the seventh column. The eighth column shows the summary data when it was used all the methods for generating. The data which have been obtained using only TetraMax scan ATPG is presented before the last column. The penultimate column shows the difference between the test obtained with TetraMax scan ATPG and final test obtained from combining all previously derived tests. The combined test detects more faults, requires fewer clock signals, but test generation takes more time. Time increase in the number of times and the decrease in the percentage of clock cycles and undetected faults are highlighted in the last column.

The dynamics of the transition fault coverage percentage are shown in Table 7. The contribution of the different methods for fault coverage is very uneven. This leads to the idea that it makes sense to supplement the test in different ways. Third line from the end shows how many faults can be detected at speed. Quantity of faults detected by the scan only on some circuits remains negligible. The influence of each method to the final result is shown in Fig. 5. The left column shows the percentage of fault coverage for the scan ATPG for each benchmark. Columns on the right show how the percentage of fault coverage is increasing using non-scan ATPG, functional test, random generation, the addition of the test and scan ATPG. At speed testing allows to detect more defects. This is an important advantage of manner proposed.

Table 6. *The experimental results with the benchmarks.*

Bench.	Param.	Sequen. ATPG	Funct.	Rand.	Similar.	Add. Scan	Total	Scan ATPG	Differ.	Ratio
	Time (s)	27042	1547	859	520	14	29982	69	-29913	-434
B14	Clocks	1113	29520	9840	2820	93345	136638	302820	166182	55%
	Not detec.	10897	3281	1116	830	36	36	166	130	78%
	Time (s)	34156	38227	19993	15945	1632	109953	2021	-107932	-53
B15	Clocks	642	48650	39200	22593	609293	720378	953676	233298	24%
	Not detec.	35031	15521	10617	9160	1218	1218	1817	599	33%
	Time (s)	54396	104564	58804	29868	3532	251164	3717	-247447	-67
B17	Clocks	101	52924	69892	36562	5439260	5598739	6042050	443311	7%
	Not detec.	105611	88991	68641	59908	8316	8316	8688	372	4%
	Time (s)	134556	32416	4460	2060	9	173501	35	-173466	-4956
B20	Clocks	1313	124432	107262	11009	131810	375826	892290	516464	20%
	Not detec.	42370	10613	782	623	234	234	2659	2425	91%
	Time (s)	205124	44904	9451	7716	43	257238	352	-266886	-758
B22	Clocks	243	133421	113322	19897	462315	729198	1749300	1020102	58%
	Not detec.	58186	15075	2125	1621	358	358	1566	1208	77%
	Time (s)	175504	29172	110497	11001	58493	384667	57904	-326763	-564
CPU	Clocks	7583	13635	3737	505	5074030	5099490	8141770	3042280	37%
	Not detec.	61177	59702	58053	57912	12035	12035	14270	2235	15%

Table 7. *Dynamics of the transition fault coverage percentage.*

Benchmarks	B14	B15	B17	B20	B22	CPU
Number of faults	21282	36230	107724	43808	65888	139274
Sequential ATPG	48,80%	3,31%	1,96%	3,28%	11,70%	56,07%
+ Functional test	84,58%	57,16%	17,39%	75,77%	77,12%	57,13%
+ Random gen.	94,75%	70,70%	36,28%	98,21%	96,77%	58,32%
+ Similar gen.	96,56%	74,71%	44,37%	98,58%	97,54%	58,42%
+ Scan ATPG	99,83%	96,64%	92,28%	99,47%	99,46%	91,36%
Only scan ATPG	99,22%	94,98%	91,93%	93,93%	97,70%	89,75%

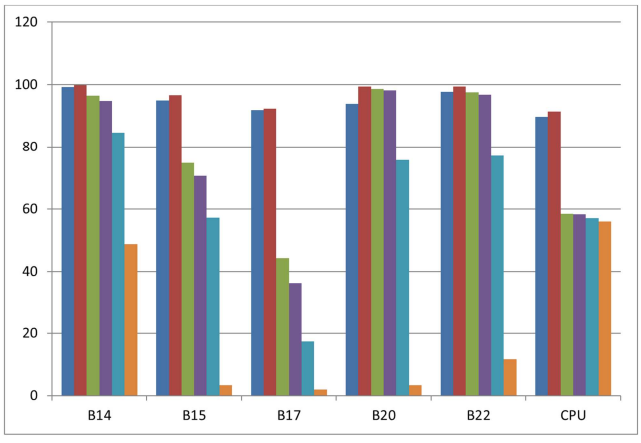

Fig. 5. *The influence of each method to the final test coverage.*

8. Conclusions

High fault's coverage cannot be achieved by one method. The proposed design flow provides the gradual addition of the test using a variety of methods. Non-scan structural test generation (ATPG) for the circuits with scan enables to detect more faults, because additional inputs and outputs of the DFT are used. Effective use of fault simulation enables to to achieve higher fault coverage.

Full scan enables the detection of the most faults of circuits. However, the full scan does not create preconditions for at speed testing. Testing at least part of the faults at speed is always meaningful, since it enables the detection of more defects. Modern non-scan ATPG tools detect a relatively small number of faults, even for circuits with embedded scan.

Methods for generating functional test can significantly complement the fault coverage. Fault simulation acceleration through faults lists of test patterns helps increase the fault coverage using only the random generation and selection based on fault simulation. Generating test patterns that were similar to those that already detected some circuit faults allows replenishment the test. It is thus intended to have a test that can detect more faults at speed. Conventional full-scan test is generated for the remaining not detected faults. These provide the higher fault coverage and at the same time the possibility of more faults to test at speed.

Experiments with benchmarks confirmed that non-scan test enables the detection of more faults, reduces the amount of clock signals required for the execution of the test, but the test generation takes significantly more time. The proposed approach is meaningful in the case where very high requirements are raised to test quality.

Non-scan test can detect a considerable proportion of fault models. Influence for fault coverage of different non-scan test generation techniques is very diverse, and therefore it makes sense to use a wide range of techniques that complement the fault coverage. Presented dynamics of the transition fault test coverage percentage demonstrates the complementarity of different methods and provide the basis for selection of test generation strategy.

The proposed circuit design scheme allows the use of modern ATPG tools and accumulated experience. The proposed way to create the fault lists for individual test patterns creates preconditions widely use fault simulation during test generation. Individual faults list for each test pattern further reduces fault simulation time and retains the ability to exploit existing fault simulation speedups. In this case, the opportunity arises to use the achievements obtained by generating the input sequence of functional test and at the same rely on fault simulation.

The proposed new generation method of similar test patterns allows the detection of new faults in the final phase of the test generation, when all the possibilities of other approaches are exhausted.

Experiments with benchmarks demonstrated that the stepwise addition of the test leads to far better fault detection, showed an increase in the degree of quality, and the impact on the quality of the final test of a variety of test generation approaches.

Embedded systems can have functions that are implemented as hardware or software. Replacement of the input and output variables of the software with binary input and output vectors enables the same hardware test generation techniques use for the software test generation. This is the object of further study.

References

[1] Crouch A. L, Potter J. D. (2016) “Invited - A box of dots: using scan-based path delay test for timing verification”. In Proceedings of the 53rd Annual Design Automation Conference (DAC'16). ACM, New York, NY, USA, Article 174, 6 pages. DOI: 10.1145/2897937.2905001.

[2] Zhang G. L, Li B, Schlichtmann U. (2016). “EffiTest: efficient delay test and statistical prediction for configuring post-silicon tunable buffers”. In Proceedings of the 53rd Annual Design Automation Conference (DAC'16). ACM, New York, NY, USA, Article 60, 6 pages. DOI: 10.1145/2897937.2898017.

[3] Niermann T, and J. H. Patel J. H. (1991) “*HITEC:* a test generation package for sequential circuits”. DATE, pp. 214-218.

[4] Hsiao M. S, Rudnick E. M and Patel J. H. (1997) “Sequential circuit test generation using dynamic state traversal”. DATE, pp. 22-28.

[5] Lin X, Pomeranz I and Reddy S. M. (1998) “MIX: a test generation system for synchronous sequential circuits”. VLSI Design Conf, pp. 456-463. DOI: 10.1109/ICVD.1998.646649.

[6] Giani A, Sheng S, Hsiao M. S, and Agrawal V. (2001) “Efficient spectral techniques for sequential ATPG”. DATE, pp. 204-208. DOI: 10.1109/DATE.2001.915025.

[7] Mainak B, Rahagude N, and Hsiao M. S. (2011) "Design-for-test methodology for non-scan at-speed testing". Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011. IEEE, DOI: 10.1109/DATE.2011.5763041.

[8] Bareisa E, Jusas V, Motiejunas K, Seinauskas R. (2013) “Delay fault testing using partial multiple scan chains”. Microelectronics reliability, Vol. 53, iss. 12. pp. 2070-2077. DOI: 10.1016/j.microrel.2013.07.002.

[9] Hashempour H, Meyer F. J and Lombardi F. (2002) “Test time reduction in a manufacturing environment by combining BIST and ATE”. Defect and Fault Tolerance in VLSI Systems, pp 186-194. DOI: 10.1109/DFTVS.2002.1173515.

[10] Chen M and Mishra P. (2010) “Functional Test Generation Using Efficient Property Clustering and Learning Techniques”. IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 29, no. 3, pp. 396-404. DOI: 10.1109/TCAD.2010.2041846.

[11] Levendel Y. H and Menon P. R.(1982) “Test generation algorithms for computer hardware description languages”, IEEE Trans. Comput., vol. 31, pp. 557-588. DOI: 10.1109/TC.1982.1676054.

[12] Mishra P, Dutt N. D. (2005) “Functional Coverage Driven Test Generation for Validation of Pipelined Processors”. In Proc. Design Automation and Testing in Europe, pp. 678-683, DOI: 10.1109/DATE.2005.162.

[13] Frühwirth T, (1998) “Theory and practice of constraint handling rules”, “The Journal of Logic Programming” Volume 37, Issues 1–3, pp. 95–138.

[14] Moskewicz M. W, Madigan C. F, Zhao Y, Zhang L, Malik S. (2001) “Chaff: Engineering an Efficient SAT Solver”. In Proceeding of the 38th Design Automation Conference, pp. 530-535 DOI: 10.1109/DAC.2001.

[15] Bareisa E, Jusas V, Motiejunas K, Seinauskas R. (2013) “Functional delay test generation approach using a software prototype of the circuit”. Computer Science and Information Systems, Vol. 10, iss. 3. p. 1165-1184.

[16] Wang L. T, Chang Y. W, and Cheng K. T. (2009) “Electronic Design Automation: Synthesis, Verification, and Test (Systems on *Silicon)”*. Morgan Kaufmann, p. 933.

[17] Parreira A, Teixeira J. P, and Santos M. (2003) "A novel approach to FPGA based hardware fault modeling and simulation". In *Proc. 6th IEEE Int. Workshop Des. Diagnostics Electron. Circuits Syst.*, pp. 17–24, DOI: 10.1.1.1.3581.

[18] Ravikumar C. P, Jain V, Dod A. (1997) "Faster Fault Simulation through Distributed Computing". *International Conference on VLSI Design,* pp. 482-487. DOI: 10.1109/ICVD.1997.568181.

[19] Lee H. K, Ha D. S. (1996) "HOPE: An Efficient Parallel Fault Simulator for Synchronous Sequential Circuits". lEEE Transactions on computer-aided design of integrated circuits and systems, Vol. 15, No. 9, pp. 1048-1058. DOI: 10.1109/43.536711.

[20] Niermann T. M, Cheng W. T, Patel J. H. (1992) "PROOFS: A Fast, Memory-Efficient Sequential Circuit Fault Simulator". lEEE Transactions on computer-aided design of integrated circuits and systems, Vol. 11, No. 2, pp. 198-207. DOI: 10.1109/DAC.1990.114913.

[21] Seinauskas, R., Seinauskas, V. (2013). "Examination of the possibilities for integrated testing of embedded systems". American Journal of Embedded Systems and Applications, Vol. 1, No. 1, pp. 1-12.

[22] Corno, F., Reorda, M. S., & Squillero, G. (2000). "RT-level ITC'99 benchmarks and first ATPG results". *IEEE Design & Test of computers*, (3), 44-53.