INFORMATICA, 2005, Vol. 16, No. 1, 19–36

The Realization-Independent Testing Based on the Black Box Fault Models

Eduardas BAREIŠA, Vacius JUSAS, Kęstutis MOTIEJŪNAS, Rimantas ŠEINAUSKAS

Software Engineering Department, Kaunas University of Technology
Studentų 50–406, LT-51368 Kaunas, Lithuania
e-mail: kestas@soften.ktu.lt

Received: June 2004

Abstract. The design complexity of systems on a chip drives the need to reuse legacy or intellectual property cores, whose gate-level implementation details are unavailable. In this paper we consider the realization-independent testing and the impact of circuit realization on the fault coverage. We investigated two fault models (input-output pin pair fault and input-input-output pin triplet fault) that are used by test generation for circuits described at system description level. The test generation on the system-level model is preferable if the efforts and the duration of the test supplement activities are less than the efforts and the duration of the test generation on gate-level model. The test set for the black-box model is larger as compared to the test set for the particular realization of the circuit. However, large test sets for the black-box model can be compacted by analysis not only according to the stuck-at faults, but also according to various defects for the particular realization.

Key words: digital circuits, realization-independent testing, fault models.

1. Introduction

As a result of modern technologies both the digital device density and the device complexity have steadily increased. In this turn, the reliability of electronic systems has become of increasing interest to manifold applications, such as computers, telecommunications, military, aerospace, banking industry, etc. One of the key requirements in order to obtain a necessary level of the reliability of devices is to perform an adequate testing. Due to the complexity of contemporary electronic devices, the task of testing has become hard and demands the evolution of new, more efficient test methods continuously. Furthermore, due to System on a Chip (SoC) and Deep Submicron the cost to manufacture a gate keeps dropping. On the other hand, the cost to test a gate stays the same. The test cost is becoming the largest portion of the total manufacturing cost.

Therefore complex, high-quality electronic products need a cost effective design and test flow. The design complexity drives the need to reuse legacy or intellectual property cores in SoC. High Level modules of SoC are often specified only in terms of their behavior. SoC designs rely heavily on reusable and pre-designed cores or intellectual property modules, whose gate-level implementation details are unavailable. System designers

become system architects reusing more and more proven components and their test processes. A test reuse must follow the same path. Conventional single stuck–at fault models associated with internal logical gates or their inter connections are not applicable for a test reuse. The structural defect-based testing involves no test reuse as tests are usually generated after the structural synthesis. The implementation depends on SoC manufacturing technologies and is permanently changing in a SoC lifecycle. How core vendors can provide reusable tests for new implementations? In this case the fault model of the SoC module should be realization-independent. However, can a test based on functional fault model be effective in uncovering physical defects? How is its effectiveness dependent on the synthesized structure? These are important questions, not only for the test reuse, but also due to the fact that soft cores can be synthesized by different electronic design automation systems and mapped in different cell libraries and manufacturing technologies.

Ideally, what is needed is a design, a test methodology and fault models that may enable a high-level design validation, the testability enhancement and the test generation in such a way that a high Defects Coverage (DC) is achieved. Of course, reaching this goal independently from the structural synthesis and from the manufacturing technology is impossible, as the test quality will always depend on the final physical realization. However, significant steps in this direction can be made.

The types of physical defects depend on the technology. Across various technologies, the most common types of defects during manufacturing are short-circuits, open bonds, open interconnections, bulk shorts, shorts due to scratches, shorts through dielectric, pin shorts, cracks and missing transistors. The first step in a test generation process is the choice of a suitable fault model. Ideally, the fault model should model all possible physical defects in the Circuit Under Test (CUT).

The defect-oriented testing relies on the layout-based test generation and the current–based (IDDQ) testing quite effectively used in submicron and deep submicron design. The defect-oriented test generation starts when the layout design is finished (Fig. 1).

The abstraction of the real defects is the main feature of the stuck-at fault model, which has been found effective to test chips in a non-deep submicron process. Testing techniques based on the stuck-at fault model are very popular and most effective for the post-silicon testing. The test generation for stuck-at faults starts before the layout design (Fig. 1). Tests for stuck-at faults cover real defects of the layout only for primitive (not complex) gates. In this case we are speaking about the layout-independent test generation. In general, the test generation task on gate level does not consider defects. However, some bridging or stuck-open fault models are used on the gate level as well.

Several approaches to the test pattern generation at the Register Transfer Level (RTL) have already been proposed. Most of them are usually able to generate test patterns of good quality, sometimes comparable to the gate-level Automated Test Pattern Generation (ATPG) tools. In case of the test pattern generation at the Register Transfer Level the test set is generated for all possible implementations and we are speaking about the realization-independent test generation. The test generation task can be done in parallel with the synthesis of the circuit on the gate level (Fig. 1).

The test generation for algorithms on the system-level relies on black-box fault models. In this case not only the implementation, but as well the synthesizable description of

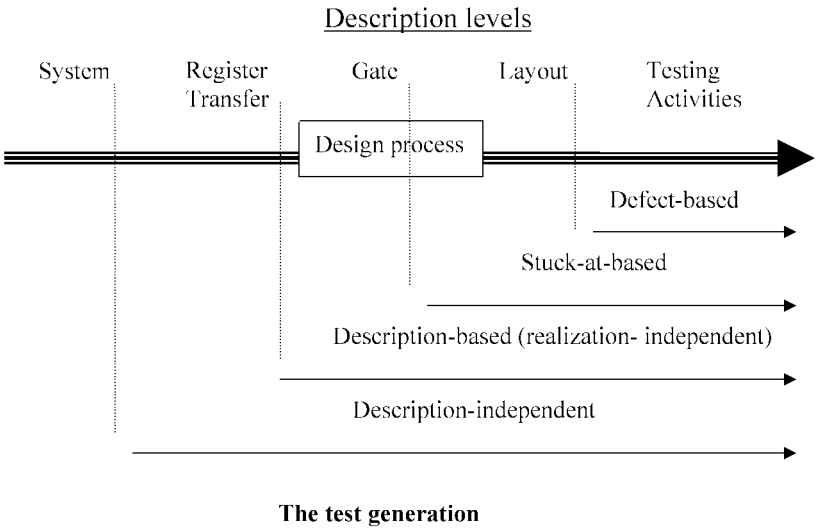

Fig. 1. The starting points of the test generation in the design process.

the behaviour is unknown. The test generation task is much more complicated, but it can be done in parallel with the design of the synthesizable description and with the synthesis of the circuit on the gate level (Fig. 1).

In this paper we consider the realization-independent testing, investigate the impact of the circuit realization on the fault coverage of the test set, the suitability of black-box fault model for the testing of different realizations and the capabilities of the test generation for the black-box model.

2. The Realization-Independent Testing

One of the concepts of the realization-independent testing is based on universal test set (Akers, 1973; Betancourt, 1971). It exploits the unateness property of a module's variables, and is composed of functionally defined minimal true and maximal false tests. A universal test set can detect both single and multiple stuck-at faults in realizations with restrictions on their structure. The size of the universal test sets is small for functions that are fully or partially unate, but it becomes exhaustive for binate functions. Expanded vectors consist of all input literals appearing in a function's minimal expression (Akers, 1973).

The universal test set of a function consists of all minimal true expanded vectors and all maximal false expanded vectors. An input vector t_i is greater than or equal to another vector t_k, denoted $t_i \geqslant t_k$, if t_i has 1 in every bit position where t_k does. The minimal true vector $\{t_i\}$ of z are input vectors such that $z(t_k) = 1$ for all $t_k \geqslant t_i$, but $z(t_k) = 0$ for all $t_k < t_i$. The maximal false vectors $\{t_i\}$ of z are input vectors such that $z(t_k) = 0$ for all $t_k \leqslant t_i$, but $z(t_k) = 1$ for all $t_k > t_i$. For example, Fig. 2 shows the realization of the

Table 1
The true table of the function F and its universal test set

Input vectors $abcd$	Expanded vectors $ab\,\bar{b}c\,\bar{c}d$	Output	Vectors of universal test set	Test type
0000	001010	0		
0001	001011	0	Test vector	Max. false
0010	001100	0		
0011	001101	0	Test vector	Max. false
0100	010010	0		
0101	010011	0	Test vector	Max. false
0110	010100	0		
0111	010101	1	Test vector	Min. true
1000	101010	1	Test vector	Min. true
1001	101011	1		
1010	101100	1	Test vector	Min. true
1011	101101	1		
1100	110010	1	Test vector	Min. true
1101	110011	1		
1110	110100	0	Test vector	Max. false
1111	110101	1		

function of Table 1. Vectors of the universal test set for a function $F(a, b, c, d)$ are derived from its true table. It has been proved that the universal test set detects all multiple stuck-at faults in a realization R under the following restriction: every path between two points in R has the same inversion parity. These realizations are called an unate gate network. The unate gate network restriction is rigorous and practically it is often replaced by a relaxed condition. The realization R of the function is a balanced inversion parity (BIP) network, if paths from unate variables have the same inversion parity. BIP realizations allow paths with different inversion parities between any binate variable and the outputs of a network, and so the restrictions are valid to any practical realization. It has been shown recently that in the worst case unate-gate networks are at most twice larger than the minimal implementation. BIP realizations, on the other hand, tend to be minimal or near-minimal. Any BIP realization can be obtained from an unate-gate network by applying a set of the resubstitution and De Morgan transformation. Therefore, the universal test set detects all detectable multiple stuck-at faults in BIP realization. The size of the universal test set for an unate function is relatively small, but for a binate function equals the size of the exhaustive test. The exhaustive test (2^n) includes all possible test vectors of n inputs.

In general case it is impossible to get a complete test set less than the exhaustive test without the information about the particular realization of the module. The complete test detects all single stuck-at faults of the module described in terms of primitive gates. It has been proved that each input vector of the true table is necessary for testing at least one realization of the function (Šeinauskas, 2003). Fig. 2 shows the realization of the

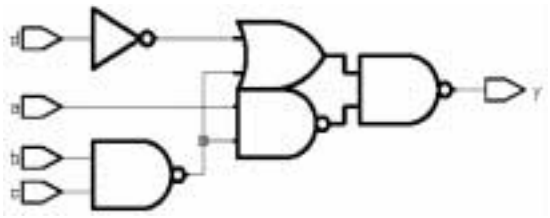

Fig. 2. The realization of the function F of Table 1.

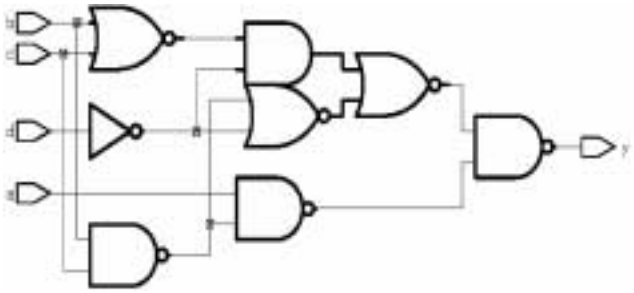

Fig. 3. The realization of the function $F1$.

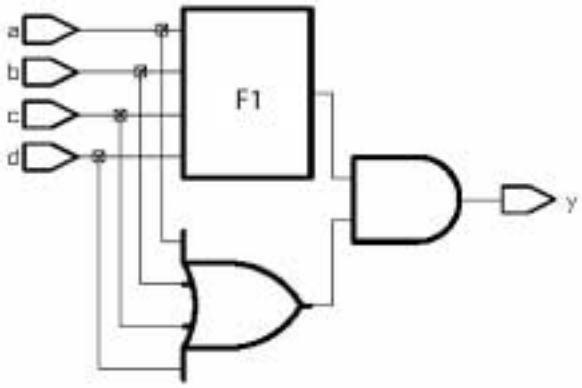

Fig. 4. A new realization of the function F.

function of Table 1. The universal test set does not include the input vector $abcd = 0000$. We can create a new realization, which requires this input vector as a test vector. Let's say, that the output of the function $F1$ differs from the output of the function F on the input vector $abcd = 0000$ and $F(0000) = 0$, $F1(0000) = 1$. Fig. 3 shows the realization of the function $F1$. The OR gate in Fig. 4 corrects the reactions of the block of the function $F1$ and we get a new realization of the function F. The test set of this realization requires input vector $abcd = 0000$. Therefore the universal test set doesn't detect all stuck-at faults, notwithstanding all of them are detectable in this realization. It should be noted, that this new realization does not meet the BIP networks restriction. The input variables

a and d of the function F are unate and the variables b and c are binate. Only the input variable a is unate for the function $F1$. Therefore, the realization of the function F in Fig. 4 contains paths with a different parity from the input of the variable d and does not fill the restriction to BIP gate networks.

A cell fault proposed in (Psarakis *et al.*, 1998) implicitly models all defects that alter a module's truth table and so provides a high degree of the realization independence. However, this model can be applied only to very small modules, because it often requires an exhaustive test set comprising all possible input vectors. The input pattern fault model (Blanton and Hayes, 1997) uses a true table of the function as well. Each input pattern fault defines an input and output pattern pair and corresponds the faulty behaviour of the module. The number of faults for even small modules is large.

Coupling faults can alter output values in response to changes occurring on one or more inputs of a function (Yi and Hayes, 2001). The simplest case is a single coupling fault, which is defined in terms of a single input/output signal pair. All test vectors for coupling faults are called as a coupling test set. The coupling test sets share some properties with universal test sets, but they are not necessarily exhaustive for binate functions. The coupling test sets are very large even for small functions.

Higher-level fault models have been proposed for the realization-independent functional testing of combinational circuits. RTL fault models and quality metrics have been considered in (Santos *et al.*, 2000). Logic/arithmetic operations, constant/ variable switch, null statements, if/else, case, for instructions has been considered as RTL fault models. In some cases, their effectiveness in covering stuck-at faults on the circuit's structural description has been ascertained. However, this does not guarantee their effectiveness to uncover physical defects or stuck-at faults.

The high-level fault models taken from the software testing have three main advantages: they are well known and quite standardized; they require little calculations, apart from the complete fault-free simulation; and they are already embedded in some commercial tools. However, while such metrics may be useful to validate the correctness of a design, they are usually inadequate to foresee the gate-level fault coverage with a high degree of accuracy. One of the most used fault models is the observability enhanced statement coverage metric proposed in (Devadas *et al.*, 1996) and (Fallah *et al.*, 1998). This fault model requires that all statements in the VHDL description are executed at least once, and that their effects are propagated to at least one primary output.

Some approaches rely on a direct examination of the HDL description (Ferrandi *et al.*, 1998) or exploit the knowledge of the gate-level implementation (Rudnick *et al.*, 1998). Some extract of the corresponding control machine (Cheng and Khrishnakumar, 1996; Moundanos *et al.*, 1996) from a behavioural description is used. The listed approaches are of limited generality and the adequacy of testing defects or of the coverage stuck-at faults on the gate level are not proved.

The behavioral view (Bareiša *et al.*, 2001; Bareiša *et al.*, 2003) or the "black-box" represents the system by defining the behavior of its outputs according to the values applied on its inputs without the knowledge of its internal organization. In this case only the input-output connectivity can be fixed (Šeinauskas and Bareiša, 1996). The connectivity fault models are rough enough compared to the stuck-at fault model. However,

the experimental investigation fault models based on input-output paths testing demonstrated a high defect and stuck-at fault coverage on the benchmark circuits (Šeinauskas and Bareiša, 1996; Jusas *et al.*, 2001; Jusas and Šeinauskas, 2002).

A n-detection test set is the one where each modeled fault is detected either by n different tests, or by the maximum number of different tests that can detect the fault if this number is smaller than n. In various types of experiments reported in (Pomeranz and Reddy, 1999a; Takahashi *et al.*, 2002), n detection test sets were shown to be useful in achieving high defect coverage for all types of circuits and for different fault models.

3. The Input-Output Pin Pair Fault Model

The different fault models based on input-output paths testing were suggested in (Šeinauskas and Bareiša, 1996; Jusas *et al.*, 2001). We provide the other presentation of the main concepts. Let the circuit have a set of inputs $X = \{x_1, x_2, \ldots, x_i, \ldots, x_n\}$ and a set of outputs $Z = \{z_1, z_2, \ldots, z_j, \ldots, z_m\}$. The pin fault model considers the stuck-at-0/1 faults occurring at the module boundary, and has a week correlation with the circuit's physical faults. We write x_i^1 and x_i^0 for the input stuck-at-1/0 faults, and z_j^1 and z_j^0 for the output stuck-at-1/0 faults. There are $2n + 2m$ possible pin faults. Input-output pin stuck-at fault pairs (x_i^t, z_j^k), $t = 0, 1$, $k = 0, 1$ are called pin pair faults (PP). The number of possible pin pair faults of the circuit is at most $4 * n * m$. We denote the set of the pin pair faults by

$$P1 = \left\{(x_i^t, z_j^k) \,|\, i = 1, \ldots, n, j = 1, \ldots, m,\ t = 0, 1,\ k = 0, 1\right\}.$$

The test vector detects the pin pair fault (x_i^t, z_j^k) of the module if the test vector detects both the pin faults x_i^t, and z_j^k of the pair on the output z_j of the module. It may appear that there exist no electric connections between the input and the output, and the pin pair fault defined by these inputs and outputs can't be detected. These faults are not testable. The PP fault (x_i^t, z_j^k) of a module is testable if a conventional deterministic (Breuer and Friedman, 1976) test generator for a realization of the module finds a test vector, which detects a pin fault x_i^t on an output z_j while the input x_i and the output z_j are set up to the $\bar{t}$ and $\bar{k}$. The number of testable PP faults equals to $4 * n * m$ minus the number of not testable PP faults. The connectivity rate demonstrates the relation between the number of testable PP faults and the total number of possible PP faults and is computed as follows:

$$\text{Connectivity rate} = \text{No. of testable PP faults}/4 * n * m$$

Note that in general it is not possible relate the PP fault with the defects of the module unambiguously, because the PP fault doesn't fix exactly the signal propagation path in the circuit. The set of the PP faults of the function F (see Table 1) includes the faults $P1 = \{(a^1, y^1),\ (a^0, y^0),\ (d^0, y^0),\ (d^1, y^1),\ (b^1, y^1),\ (b^1, y^0),\ (b^0, y^1),\ (b^0, y^0),\ (c^1, y^1), (c^1, y^0),\ (c^0, y^1),\ (c^0, y^0)\}$. The six test vectors 1010, 1110, 0011, 0111, 1100, 0101 detect all the PP faults. The test vector 1010 detects the PP faults (b^1, y^0), (a^0, y^0),

the test vector 1110 detects the PP faults (b^0, y^1), (c^0, y^1), the test vector 0011 detects the PP fault (b^1, y^1) and so on. The six test vectors detect all the stuck-at faults for the realization of the function F in Fig. 2.

The core can be synthesized by different electronic design automation systems, and mapped into different cell libraries and manufacturing technologies. An important issue is how the test set of the core covers the faults of new implementations, which are done by the same synthesizer. The ISCAS'85 benchmarks (Brglez and Fujiwara, 1985) have been selected for experiments. The original ISCAS'85 circuits have been re-synthesized with the Synopsys Design Compiler program by the default mode and by using the AND-NOT cell library of two inputs. The three realizations have been analyzed:

R1 – the non-redundant ISCAS'85 benchmark circuit

R2 – Synopsys Design Optimization, the target library – class.db (the default mode)

R3 – Synopsys Design Optimization, the target library – and_or.db

The number of stuck-at faults for each realization we can see in Table 2.

The original benchmark realizations have more stuck-at faults in total. It means that re-synthesized circuits were more optimized. The percent of the difference between maximum and minimum numbers to the maximum number of stuck-at faults varies from 9 to 53. It demonstrates the impact of the target library by the design synthesis and the diversity of realizations.

The defining of the set of PP faults for the black-box model is complicated. However, if we know the structure of the considered circuit, the deterministic test generator for stuck-at faults can be used for defining the set of testable PP faults – the PP fault (x_i^t, z_j^k)

Table 2

The number of stuck-at faults in three different realizations of the ISCAS'85 benchmarks

Circuit	R1	R2	R3	D	%
C432	507	420	460	87	16
C499	750	978	1246	496	40
C880	942	854	928	88	9
C1355	1566	1316	1406	250	16
C1908	1862	875	1224	987	53
C2670	1990	1498	1658	492	25
C3540	3126	2451	2520	675	21
C5315	5248	3875	4130	1373	26
C6288	7638	6680	7498	958	13
C7552	7039	4573	4798	2466	35
Total	30668	23520	25868		

R1 – the non-redundant ISCAS'85 benchmark circuits,
R2 – Synopsys Design Optimization, the target library – class.db,
R3 – Synopsys Design Optimization, the target library – and_or.db,
D – the difference between maximum and minimum numbers,
% – the percent of the difference to the maximum number.

is detectable, if stuck-at-t fault of the input x_i is detectable on output z_j by output response k. The comparison of the testable PP fault set size with the stuck-at fault set size of the original ISCAS'85 benchmark circuit realization and the comparison of the length of the test sets are given in Table 3. Note that the numbers of the stuck-at faults and the numbers of the PP faults are of the same rate. The test sets for the PP faults are in all cases larger than the test sets for the stuck-at faults significantly. The test set for PP faults has been got using the random search procedure (Breuer and Friedman, 1976). The test sets for stuck-at faults have been generated using the deterministic SYNOPSYS ATPG generator (Breuer and Friedman, 1976; Synopsys, 2000).

The numbers of undetected stuck-at faults of three realizations for the test sets of the PP faults for the benchmark circuits are given in Table 4. The average percent of undetected faults doesn't exceed 0.5%, but the maximum percent of undetected faults reaches 3.4%. The detecting of the PP faults by the tests generated for the stuck-at faults is presented in Table 5. The test sets have been generated for the original realization R1 by a deterministic algorithm (D) and by a random & deterministic algorithm (R&D) (Breuer and Friedman, 1976; Synopsys, 2000). The deterministic algorithm has been used if the random search did not reach a 100% fault coverage. As we see, the test sets, which detect 100% stuck-at faults of the benchmark circuits, detect on average about 60% of the PP faults. The experimental results show that the test sets, which are generated according to the PP fault model, obtain high fault coverage of gate stuck-at faults. However, the PP fault coverage of the test sets targeted for the stuck-at fault is very low. This implies that a test set based on the PP fault model covers far much more than the single stuck-at faults. It is very likely the test vectors based on the PP fault model can cover other kinds of the faults such as bridging faults, multiple stuck-at faults and stuck-at faults of different circuit realizations.

Table 3

The number of stuck-at and PP faults and its test sets sizes of of the original ISCAS'85 benchmark circuit realization

Circuit	Test generation for stuck-at faults		Test generation for PP faults	
	Stuck-at Faults	Test size	Testable PP Faults	Test size
C432	507	63	540	117
C499	750	63	5184	1077
C880	942	54	1326	381
C1355	1566	92	5184	1011
C1908	1862	123	3004	620
C2670	1990	113	3320	448
C3540	3126	172	2588	515
C5315	5248	130	10540	1169
C6288	7638	34	3068	268
C7552	7039	209	12188	2115

Table 4
The number of undetected stuck-at faults of the test sets for PP faults

Circuit	Undetected of R1	Undetected of R2	Undetected of R3
C432	11 (2.2%)	6 (1.4)%	3 (0.7%)
C499	0	0	0
C880	0	0	0
C1355	0	0	1
C1908	64 (3.4%)	9 (1.0%)	9 (0.7%)
C2670	8 (0.4%)	7 (0.5%)	6 (0.4%)
C3540	35 (1.1%)	25 (1.0%)	30 (1.2%)
C5315	0	0	0
C6288	0	0	0
C7552	25 (0.4%)	8 (0.2%)	4 (0.1%)
Total	143 (0.5%)	55 (0.2%)	53 (0.2%)

R1 – the non-redundant ISCAS'85 benchmark circuit,
R2 – Synopsys Design Optimization, the target library – class.db,
R3 – Synopsys Design Optimization, the target library – and_or.db.

Table 5
The number of detected PP faults of the test sets for stuck-at faults

Circuit	PP	D		R&D	
	faults	Detected	%	Detected	%
C432	540	406	75.13	467	86.48
C499	5184	1510	29.13	1434	27.66
C880	1326	837	63.12	862	65.01
C1355	5184	2718	52.43	2638	50.89
C1908	3004	1521	50.63	1740	57.92
C2670	3320	2204	66.38	2142	64.51
C3540	2588	2051	79.31	2036	78.73
C5315	10540	7148	67.88	7305	69.30
C6288	3068	2353	76.74	2393	77.99
Total	34754	20748	59.69	21017	60.47

D – synopsys deterministic test patterns for the realization R1,
R&D – synopsys random and deterministic test patterns for the realization R1.

A pin pair fault can concern few physical signal propagation paths between an input and an output. The testing of pin pair faults cannot guarantee the detection of all stuck-at faults of the module. The n-detection of PP fault makes sense. The change of the input value of the test pattern, which detects the PP fault (x_i^t, z_j^k), generates the adjacent test pattern (Pomeranz and Reddy, 1999b; Bareiša *et al.*, 2004), which detects the PP fault $(x_i^{\bar{t}}, z_j^{\bar{k}})$. The test size of the test sets supplemented with the adjacent test vectors

is given in Table 6. The number of the undetected stuck-at faults of three realizations for the supplemented test sets of the benchmark circuits is given in Table 6 as well. The supplemented test sets of the PP faults detect almost all stuck-at faults in all three realizations. The average percent of the undetected faults doesn't exceed 0.016%, and the maximum percent of the undetected faults reaches 0.23% only.

We see that the test sets for the PP faults and for the supplemented test sets detect less stuck-at faults for the original realization R1. It may be explained in such a way: the original circuits were synthesized early and re-synthesized by Synopsys the benchmark circuits are more optimized and have less stuck-at faults. It means that the fault coverage of the test sets generated for the black box model depends on the optimization level during the synthesis of the circuits.

In order to present the percent of the undetected faults in a uniform way for the PP faults and for the test reuse of the other realizations, the average and the maximum percent of the undetected faults for each realization pair was computed and given in Table 7. The test sets generated for the PP faults are comparable according to the average and the maximum of the undetected faults with the test reuse of the double-detection test sets for the stuck-at faults. Note that the PP test sets are generated for the black-box model of the circuits and the gate level implementation details are unavailable. The black-box model represents the system by defining the behavior of its outputs according to the values applied to its inputs without the knowledge of its internal organization. The black box models in the programming language C for ISCAS'85 benchmark circuits were used by the test generation for the PP faults.

Table 6

The number of undetected stuck-at faults of the supplemented test sets for PP faults

Circuit	Test size	Undetected of R1	Undetected of R2	Undetected of R3
C432	1086	0	0	0
C499	38161	0	0	0
C880	14672	0	0	0
C1355	35945	0	0	0
C1908	16563	4(0.21%)	2(0.23%)	0
C2670	28422	0	0	0
C3540	15055	0	0	0
C5315	108202	0	0	0
C6288	8875	0	0	0
C7552	178713	1(0.01%)	1(0.02%)	0
Total	445694	5(0.016%)	3(0.013%)	0

R1 – the non-redundant ISCAS'85 benchmark circuit,
R2 – Synopsys Design Optimization, the target library – class.db,
R3 – Synopsys Design Optimization, the target library – and_or.db

Table 7

The maximum and the average percent of the undetected faults of one and double detection test sets

The test sets	P_R2+R3		P_R1+R3		P_R1+R2	
	Average	Maximum	Average	Maximum	Average	Maximum
One-detection stuck-at fault test sets	0.75	5.50	1.27	6.66	1.34	4.92
Double detection stuck-at fault test sets	0.19	2.20	0.65	4.21	0.59	2.92
One-detection PP fault test sets	0.21	0.78	0.49	2.69	0.46	2.25
Double detection PP fault test sets	0.09	0.28	0.20	0.63	0.21	0.66

P_Ri+Rj – the percent of the undetected faults of two realizations Ri and Rj.

4. The Input-Input-Output Pin Triplet Fault Model

The PP fault model requires the path sensitisation between an input and an output at least one time. The sensitisation of the paths pair would increase the rate of the separate paths sensitisation. The pin triplets contain such a property.

The input-input-output pin stuck-at fault triplets (x_i^t, y_h^p, z_j^k), $t = 0,1$, $k = 0,1$, $p = 0,1$ are called the pin triplets faults (PT). The number of possible pin triplets faults of the circuit is at most $4 * n * n * m$. We denote the set of the pin triplet's faults by

$$P1 = \big\{(x_i^t, y_h^p, z_j^k)| \\ i = 1,\ldots,n,\ h = 1,\ldots,n,\ j = 1,\ldots,m,\ t = 0,1,\ k = 0,1,\ p = 0,1\big\}.$$

The test vector detects the pin triplet fault (x_i^t, y_h^p, z_j^k) of the module if the test vector detects the pin faults x_i^t, y_h^p and z_j^k of the triplet on the output z_j of the module. The pin triplet fault requires the sensitisation of two paths from inputs to the output on the same test vector. All possible pairs of the sensitisation paths will be considered.

The PT fault covers the PP fault if x_i and y_h is the same input. The pin stuck-at fault triplet (x_i, x_i, z_j) corresponds to the pin stuck-at fault pair (x_i, z_j). The test sets for the PT faults were computed using the black-box model and the random search. The test size and the undetected stuck-at faults for three realizations are given in Table 8. The number of the detected PT faults is provided as well.

We see that the test sets for the input-input-output pin stuck-at fault triplets almost covers stuck-at faults for all realizations. The generated test sets are suitable for all realizations and the size of the test sets is huge. The necessary test pattern for each realization can be selected from the generated test sets by a fault simulation and the number of the selected test pattern is given in Table 8 as well.

Table 8

The number of undetected stuck-at faults of the test sets for PT faults

Circuit	Test size	PT faults	UnR_1	NTR_1	UnR_2	NTR_2	UnR_3	NTR_3
C432	1123	15254	0	77	0	67	0	69
C499	3410	412736	0	65	0	91	0	103
C880	4954	55280	0	73	0	49	0	53
C1355	3356	412736	0	105	0	109	0	109
C1908	2505	154284	7 (0.38%)	163	3 (0.34%)	94	3 (0.25%)	124
C2670	3259	187270	0	173	0	161	0	165
C3540	7022	123332	0	188	0	144	1 (0.04%)	149
C5315	4679	269726	0	207	0	175	0	174
C6288	2178	152802	0	59	0	73	0	72
C7552	14310	805932	6 (0.09%)	346	1 (0.02%)	266	0	238
Total	46796	2589352	13	1456	4	1229	4	1256

NTR_i – the number of the test patterns selected by a fault simulation for the realization R_i,
UnR_i – the number of the undetected stuck-at faults for the realization R_i.

Table 9

The sizes of test sets generated for stuck-at faults and sets selected from PT fault test

Circuit	Test sets generated for stuck-at faults			Test sets selected from PT fault test		
	R1	R2	R3	R1	R2	R3
C432	57	46	45	77	67	69
C499	54	74	80	65	91	103
C880	62	49	50	73	49	53
C1355	86	83	80	105	109	109
C1908	118	57	75	163	94	124
C2670	105	120	116	173	161	165
C3540	167	143	147	188	144	149
C5315	130	99	89	207	175	174
C6288	43	47	34	59	73	72
C7552	211	146	138	346	266	238
Total	1033	864	854	1456	1229	1256

The sizes of the tests, generated deterministic and selected by a fault simulation from the test sets, which detect the PT faults is given in Table 9 for the comparison. We see that the test sets generated for the black box is only about one and a half times longer.

We will demonstrate by a simple example why a test set for PP and PT faults doesn't guaranty the detecting of stuck-at faults. The circuit of the example is given in Fig. 5. The circuit has one fan-out with branches b_1 and b_2. All possible input vectors are listed

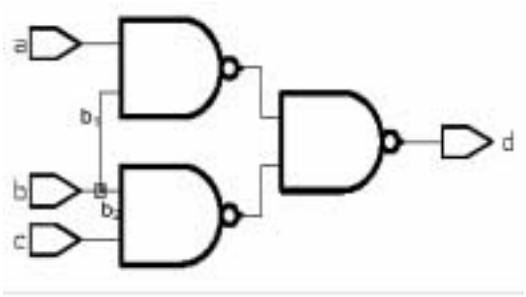

Fig. 5. The example circuit.

Table 10
The true table of the example circuit and on the input vector detectable faults

No.	a	b	c	d	PP faults	Fan-out faults	PT faults
0	0	0	0	0			
1	0	0	1	0	(b^1, d^1)	b_2^1	(b^1, b^1, d^1)
2	0	1	0	0	(c^1, d^1), (a^1, d^1)		(a^1, c^1, d^1)
3	0	1	1	1	(b^0, d^0), (c^0, d^0)	b_2^0	(b^0, c^0, d^0)
4	1	0	0	0	(b^1, d^1)	b_1^1	(b^1, b^1, d^1)
5	1	0	1	0	(b^1, d^1)	b_1^1 b_2^1	(b^1, b^1, d^1)
6	1	1	0	1	(b^0, d^0), (a^0, d^0)	b_1^0	(a^0, b^0, d^0)
7	1	1	1	1	(b^0, d^0)		

in Table 10, the PP faults, the fan-out stuck-at faults and the PT faults detectable on the input vector.

The input vectors 2, 3 and 6 will be always selected by detecting PP or PT faults. The (b^1, d^1) PP fault can be detected by one of the three vectors 1 or 4 or 5. In case of a selection of the input vector 5 all fan-out stuck-at faults will be detected. In case of selecting of the input vector 1 or 4 some stuck-at fault of the fan-out remains undetected. Only both vectors 1 and 4 can guaranty the detection of all stuck-at faults of the fan-out. The PP fault model and the PT fault model can never secure the selection of both vectors 1 and 4 or the selection of the vector 5.

Despite the drawback of the fault models demonstrated above, the test sets generated according to the mentioned models for whole benchmark circuits detect stuck-at faults of any realization unexpected well. We see a few reasons why the results are surprisingly good. First of all, functions of the different outputs depend on the same inputs. One stuck-at fault can be detected on several outputs. The test pattern for one input can detect stuck-at faults for other outputs. Actually, each test pattern can detect several PP faults and as result each PP fault may be tested more than once.

We analyzed the parameters of the circuits in order to highlight what has the impact on the stuck-at faults coverage of the black-box test sets. The connectivity and output/input rate, the number of all circuit paths (Pomeranz and Reddy, 1996) and the summary num-

Table 11

The parameters of the circuits

Circuit	Inputs n	Outputs m	$4*n*m$	Testable PP faults	Connectivity =testable connection/ $4*n*m$	(O/I)*100 rate%	Number of the paths	Undetected stuck-at faults
C432	36	7	1008	540	50%	19.4	–	20
C499	41	32	5248	5184	99%	78.0	–	0
C880	60	26	6240	1326	21%	43.3	17264	0
C1355	41	32	5248	5184	99%	78.0	8346432	1
C1908	33	25	3300	3004	91%	75.8	1458114	88
C2670	157	64	40192	3320	8.3%	40.8	1359768	21
C3540	50	22	4400	2588	59%	44.0	57353342	90
C5315	178	123	87576	10540	12%	69.1	2682610	0
C6288	32	32	4096	3068	75%	100	$\sim 10^{20}$	0
C7552	206	107	88168	12188	12%	50	1452986	39

ber of the undetected stuck-at faults (all realizations) of the test sets for the PP faults and of the supplemented test sets are given in Table 11.

In Table 11 we don't see the definite impact of the circuit parameters on the number of the undetected faults for the test sets generated for the black box model. We see only the correlation of the connectivity rate and the output/input rate of the circuits. It is likely that the complexity of the implemented functions is playing the fundamental role for the number of the undetected faults.

5. Conclusions

The design for the test and the test generation on system-level model reduces time-to-market. The test generation on the system-level model can not guarantee complete fault coverage on the gate-level model for each possible implementation. The test generation on the system-level model is preferable if the efforts and the duration of the test supplement activities are less than the efforts and the duration of the test generation on gate-level model. The experiments show that the test sets generated for black-box faults at system level detect in average more than 99 percent of the stuck-at faults of the three different circuit realizations at gate level. The test sets supplemented with the adjacent test vectors declined both the maximum and the average percent of undetected faults more than twice.

In general, the test generation task for the black-box model is more complicated, because possible realizations of the design must be taken into account. Therefore, the test set for the black-box model is larger as compared to the test set for the particular realization of the circuit. However, the time for the test generation of the black box model is not so critical, because the test generation can be done in parallel with the circuit synthesis process without a prolongation of Time-to-Market. Large test sets for the black-

box model can be compacted by analysis not only according to the stuck-at faults, but also according to various defects for the particular realization.

The test generation approach consisting of the test generation for the black-box model and of the test set compaction according to the defects of the particular realization can be used. We are aware that 100% stuck-at fault tests cannot achieve a 100% defect coverage. The test generation for the defects is very time consuming task. We believe that the test sets generated for the black-box model detect more defects than the pure stuck-at fault test sets. As we have said above, the test sets for the black-box model can be generated with different level of accuracy. The impact of test generation methods for the black-box model and the coverage of various types of defects for different realizations can be investigated in near future.

References

Akers, S.B. (1973). Universal test sets for logic networks. *IEEE Trans. Computers*, **C-22**, 835–839.

Bareiša, E., K. Motiejūnas, R. Šeinauskas (2001). A method for exact identifying of undetectable faults in synchronous sequential circuits. *Information Technology and Control*, **2**(19), 7–20.

Bareiša, E., K. Motiejūnas, R. Šeinauskas (2003). Identifying legal and illegal states in synchronous sequential circuits using test generation. *Informatica*, **14**(2), 135–154.

Bareiša, E., V. Jusas, K. Motiejūnas, R. Šeinauskas (2004). The influence of circuit re-synthesizing on the fault coverage. *Information Technology and Control*, **2**(31).

Betancourt, R. (1971). Derivation of minimum test sets for unate logic circuits. *IEEE Trans. Computers*, **C-20**, 1264–1269.

Blanton, R.D., J.P. Hayes (1997). Properties of the input pattern fault model. In *Proc. of 1997 International Conference on Computer Design*. pp. 372–380.

Breuer, M.A., A.D. Friedman (1976). *Diagnosis & Reliable Design of Digital Systems*. Computer Science Press.

Brglez, F., and H. Fujiwara (1985). A neutral netlist of 10 combinational benchmark circuits and a target translator in Fortran. In *IEEE Int'l Symp. on Circuits and Systems*, Vol. 3. pp. 663–698.

Cheng, K.-T., A.S. Khrishnakumar (1996). Automatic generation of functional vectors using the extended finite state machine model. *ACM Trans. on Design Automation of Electronic Systems*, **1**(1), 57–79.

Devadas, S., A. Ghosh, K. Keutzer (1996). An observability-based code coverage metric for function simulation. In *Proceedings/ACM international Conference on Computer Aided Design*. pp. 418–424.

Fallah, F., S. Devadas, K. Keutzer (1998). OCCOM: efficient computation of observability-based code coverage for functional verification. In *Proceedings 35th Design Automation Conference*. pp. 152–157.

Ferrandi, F., F. Fummi, D. Sciuto (1998). Implicit test generation for behavioral VHDL models. In *Proceedings IEEE International Test Conference*. pp. 587–596.

Jusas, V., K. Paulikas, R. Šeinauskas (2001). Procedures for selection of validation vectors on the algorithm level. In *2nd IEEE Latin-American Test Workshop*. Cancun, Mexico. pp. 90–95.

Jusas, V., R. Šeinauskas (2002). Automatic test patterns generation for simulation-based validation. In *Proc. of the 8-th Biennal Baltic Electronics Conference*. Tallinn Technical University, Tallinn. pp. 295–299.

Moundanos, D., J.A. Abraham, Y.V. Hoskote (1996). A unified framework for design validation and manufacturing test. In *Proceedings IEEE International Test Conference*. pp. 875–884.

Pomeranz, I., and S.M. Reddy (1996). On the number of tests to detect all path delay faults in combinational logic circuits. In *IEEE Transaction on Computers*, **45**(1), 50–62.

Pomeranz, I., and S.M. Reddy (1999a). On n-detection test sets and variable n-detection test sets for transition faults. In *Proc. 17th VLSI Test Symp.* pp. 173–179.

Pomeranz, I., and S.M. Reddy (1999b). Pattern sensitivity: a property to guide test generation for combinational circuits. In *Proc. 8th Asian Test Symposium*. pp. 75–80.

Psarakis, M., D. Gizopoulos and A. Paschalis (1998). Test generation and fault simulation for cell fault model using stuck-at fault model based test tools. *Journal of Electronic Testing*, **13**, 315–319.

Rudnick, E.M., R. Vietti, A. Ellis, F. Corno, P. Prinetto, M. Sonza Reorda (1998). Fast sequential circuit test generation using high-level and gate-level techniques. In *Proceedings IEEE European Design Automation and Test Conference*. pp. 570–576.

Santos, M.B., F.M. Goncalves, I.C. Teixera and J.P. Teixera (2000). RTL-function test generation for high defects coverage in digital SOCs. In *Proc. of the IEEE European Test Workshop, ETW'00*. pp. 99–104.

Synopsys. Inc., 700 East Middlefield Rd., Mountain View, CA 94043. TetraMAX ATPG User Guide, v2000.11 edition. November 2000. Document Order Number: 370043-000 TBD.

Šeinauskas, R., E. Bareiša (1996). Test selection based on the evaluation of input stuck-at faults transmissions to output. *Information Technology and Control*, **2**(3), 15–18.

Šeinauskas, R. (2003). ASIC design flow and test generation. *Radioelectronics and Informatics*, **3**(24), 74–76.

Takahashi, H., K.K. Saluja, Y. Takamatsu (2002). An alternative method of generating tests for path delay faults using ni-detection test sets. In *Proc. of the 2002 Pacific Rim International Symposium on Dependable Computing* (*PRDC'02*). pp. 275–282.

Yi, J., and J.P. Hayes (2001). A fault model for function and delay testing. In *Proc. of the IEEE European Test Workshop, ETW'01*. pp. 27–34.

E. Bareiša graduated from Kaunas Polytechnic Institute in 1987. Currently he is in position of assoc. professor at Software Engineering Department, Kaunas University of Technology, Lithuania. His research interests include high-level synthesis and VLSI test generation.

V. Jusas graduated from Kaunas Polytechnic Institute in 1982. Currently he is in position of assoc. professor at Software Engineering Department, Kaunas University of Technology, Lithuania. His research interests include VLSI test generation at various levels of abstraction.

K. Motiejūnas graduated from Kaunas Polytechnic Institute in 1981. Currently he is in position of assoc. professor at Software Engineering Department, Kaunas University of Technology, Lithuania. His research interest is test generation for digital circuits.

R. Šeinauskas graduated from Kaunas Polytechnic Institute in 1972. He got his Doctor habilitus from Leningrad Energetics Institute in 1982. Currently he is in position of professor at Software Engineering Department, Kaunas University of Technology, Lithuania. His research interest is VLSI test generation.

Nepriklausomas nuo realizacijos testavimas, besiremiantis juodos dėžės gedimų modeliais

Eduardas BAREIŠA, Vacius JUSAS, Kęstutis MOTIEJŪNAS, Rimantas ŠEINAUSKAS

Straipsnyje analizuojamas nepriklausomas nuo realizacijos skaitmeninių schemų testavimas. Autoriai pasiūlė du naujus gedimų modelius, skirtus testų generavimui schemoms, kurios yra aprašytos sisteminiame lygmenyje. Eksperimentai parodė, kad testiniai rinkiniai, sudaryti remiantis juodos dėžės gedimų modeliais aptinka vidutiniškai daugiau nei 99 procentus skirtingų realizacijų schemos konstantinių gedimų.