Computing and Informatics, Vol. 32, 2013, 251–272

COMPRESSED SKEWED-LOAD DELAY TEST GENERATION BASED ON EVOLUTION AND DETERMINISTIC INITIALIZATION OF POPULATIONS

Roland Dobai, Marcel Baláž

Institute of Informatics
Slovak Academy of Sciences
Bratislava, Slovak Republic
e-mail: {roland.dobai, marcel.balaz}@savba.sk

Abstract. The current design and manufacturing semiconductor technologies require to test the products against delay related defects. However, complex system-on-chips (SOCs) require low-overhead testability methods to keep the test cost at an acceptable level. Skewed-load tests seem to be the appropriate way to test delay faults in these SOCs because the test application requires only one storage element per scan cell. Compressed skewed-load test generator based on genetic algorithm is proposed for wrapper-based logic cores of SOCs. Deterministic population initialization is used to ensure the highest achievable transition delay fault coverage for the given wrapper and scan cell order. The developed method performs test data compression by generating test vectors containing already overlapped test vector pairs. The experimental results show high fault coverages, decreased test lengths and better scalability in comparison to recent methods.

Keywords: Genetic algorithms, automatic test pattern generation, test data compression, test length, transition delay fault, skewed-load

Mathematics Subject Classification 2010: 94C12, 68M15

1 INTRODUCTION

The current design and manufacturing semiconductor technologies result in increased occurrence of delay related defects. These defects have to be covered by the test in order to ensure the high quality product requirements [2].

Delay defects are modeled by delay faults and can be detected by application of vector pairs. The first vector of the pair is the initialization vector and the second one is the excitation vector. The initialization vector sets the required value in the given fault location and the excitation vector excites and propagates the fault to an observable point in the following clock cycle. The application of arbitrary vector pairs in sequential circuits requires scan cells consisting of two memory elements because the initialization vector needs to be applied continuously to the circuit while the excitation vector is shifted in. This additional area overhead is high and therefore usually unacceptable, especially for complex system-on-chips (SOCs). Logic cores of SOCs surrounded by wrappers need wrapper boundary registers (WBRs) also with two memory elements per cell [4].

Skewed-load [22] and broadside [24] tests ensure low-overhead delay fault testability of sequential circuits. The excitation vector does not need to be shifted in but is derived from the initialization vector. The excitation vector for skewed-load test is created by one-bit shift of the initialization vector, and for broadside test it is the response of the circuit to the initialization vector. Skewed-load tests seem to be the appropriate low-overhead way to test delay faults in logic cores of SOCs because the test application requires only one storage element per cell in the WBR and scan chain, and extensive core partitioning is not required (to support broadside tests). Skewed-load tests usually achieve lower fault coverages than tests consisting of arbitrary vector pairs. The reason is the shift dependence of the excitation vector on the initialization vector [4].

Several delay fault models are known, e.g. transition delay fault (TDF) model, or path delay fault (PDF) model. TDF assumes an increased delay on the given circuit line which causes the capture of an incorrect value. Two types of faults are considered on every circuit line: slow-to-raise (STR) and slow-to-fall (STF). The initialization vector should set the circuit line to `0` (logic false) for the STR fault and the excitation vector should detect the stuck-at-`0` fault on that line in the next cycle. Similarly, the STF is detected by initializing the line to `1` (logic true) and covering the stuck-at-`1` fault in the next cycle [4, 7].

Genetic algorithms as the subclass of evolutionary algorithms proved to be powerful optimization tools in different areas such as electrical engineering, very large scale integration systems, image processing [25, ch. 10] and also circuit reconfiguration for test data reduction [26, 27]. They perform very well for difficult and wide scale optimization problems. Genetic algorithms imitate the process of natural selection: more capable individuals have better chance to succeed in life. Genetic algorithm works with a set of individuals (population) which are the possible solutions for the given optimization problem. Each individual is characterized and described by a chromosome. The ability of the individual to be successful in solving the problem is measured by its fitness. The individual with better fitness has higher probability to survive and to reproduce. Elitism can be used to ensure the unconditional survival of the fittest individuals. New individuals are created by reproduction which is made possible by several operators: selection, crossover, mutation. The selection facilitates the reproduction of individuals based on their fitness.

A fitter individual is selected for breeding always with higher probability. Crossover is executed over the chromosomes of the selected individuals; their characteristics are inherited to the offspring. The chromosome of the offspring will be based on the parents' chromosomes but also some additional (small) changes are performed by means of mutation [25].

The main advantage of the skewed-load test, as mentioned before, is a low overhead; on the contrary, its disadvantage is the shift dependence which can cause low fault coverage. The work presented in this paper utilizes the advantage of low overhead and tries to eliminate the possible lower fault coverage together with simultaneous reduction of the overall test application time. Compressed skewed-load test pattern generator (TPG) based on genetic algorithm (SKEWGEN) for wrapper-based logic cores of SOCs is proposed in this paper (it will be assumed throughout this paper that test compression is vector overlapping). Deterministic initialization of populations (population assembled by deterministic test generation) is used to ensure the highest achievable TDF coverage for the given wrapper and scan cell order. The developed method performs test data compression to decrease the test data and application time by generating test vectors containing already overlapped test vector pairs.

The rest of the paper is organized as follows. Section 2 contains the related work. The test application is discussed in Section 3. SKEWGEN is proposed in Section 4 and the deterministic initialization of populations in Section 5. Section 6 summarizes the achieved results and Section 7 concludes the paper.

2 RELATED WORK

The first method which dealt with skewed-load test generation for WBRs containing only one storage element per cell was the skewed-load for wrapper (SfW) method [3]. The method is applicable to logic cores with IEEE 1500 [1] compliant WBRs. Therefore, the test area is kept low. The SfW method assembles the test based on the pre-generated set of initialization and excitation vectors. The main disadvantage is that each initialization and excitation vector needs to be available for every TDF (these vectors are redundant for the given fault and the method chooses that vector which results in a shorter test after the overlap with the vectors for the other faults). This overhead influences the test generation. This set of vectors is reduced by some heuristics and the best overlap with the highest TDF coverage is searched for. This can be time consuming since there is a huge number of possibilities, therefore the method is usable for small circuits only.

The test generation based on satisfiability (SAT) is a well explored area [11]. SAT-based TPG for PDFs was published in [13] and broadside TPG for TDFs in [14]. SAT-based test data compression was proposed in [5] and used by the skewed-load method based on satisfiability (SKEWSAT) [10] for TDFs of SOCs. SKEWSAT overcomes the limitation of the SfW method by not using a pre-generated set of vectors. The generation is bit oriented instead of vector pair oriented generation.

SKEWSAT always searches for the smallest sequence of bits to detect another TDF. This results in a high number of SAT-problems whose solving is time consuming for larger SOC cores. The method proposed in [6] is aimed to accelerate the SAT-based test compression by early identification of unsatisfiable conditions. However, the improvement is still insufficient for larger circuits.

Recent research was also conducted to make the test application of delay faults more effective [16] and to combine skewed-load and broadside tests to achieve better fault coverage [17–21]. These methods are significantly different from the work presented in this paper because they do not consider wrapper-based SOCs where the primary inputs (PIs) and the primary outputs (POs) are accessible through WBRs, i.e. they assume that the PIs and the POs are directly accessible from outside.

Other methods have been developed to improve the TDF coverage of skewed-load tests [15, 23]. However, the improvements are limited because the shift dependence was not resolved. The scan reordering method based on the number of untestable paths proposed in [9] can be time consuming since the selection is based on TPG. The method in [28] is based on the analysis of the test set and its effect is influenced by the number of undefined values in the test (for good results it requires high number of undefined values). The method published in [8] is based on structure analysis and therefore it is independent of the test set. Two pseudo-primary inputs (PPIs) can be connected to neighboring scan cells if they do not have any common combinational successors (logic gates reachable from both PPIs). If there are only PPIs with common combinational successors then the one with the least number of reachable outputs is selected.

SKEWGEN proposed in this paper represents another important step in delay test generation for SOCs because with small area overhead and reduced test data it ensures the highest achievable fault coverage. SKEWGEN always tries to detect the highest number of new TDFs with each test vector, while the previous methods targeted one TDF only. Consequently, less amount of test data will be required and the test will be shorter. SKEWGEN uses deterministic initialization of populations, therefore the test data reduction is possible without TDF coverage loss. Usually, similar methods first generate the test vector pairs and consequently they compact them (by overlapping). SKEWGEN uses a different, reverse approach. It generates vectors directly, and the vector pairs are assembled from the generated vectors in order to perform fault simulation.

3 TEST APPLICATION

The proposed new SKEWGEN method generates tests for logic cores of SOCs with only one required memory element per cell in the WBR and in the internal scan chain. Figure 1 shows an example of wrapper configuration with only the relevant inputs, outputs and internal blocks. The architecture is IEEE Standard 1500 [1] compatible. The WBRs are constructed using WC_SD1_CII [1] cells containing one memory element in shift path dedicated to wrapper function. The generated test

is shifted in through a wrapper parallel input (WPI) labeled WPI_0. This shift-in path consists of a WBR labeled WBR_0 containing $n - s$ cells and an internal scan chain containing s cells, where n is the total number of inputs of the combinational part of the core under test. The shift-in path is concluded by a wrapper parallel output (WPO) labeled WPO_0. The response of the core is shifted out for evaluation through the path between WPI_1 and WPO_1. This path contains another WBR labeled WBR_1 comprising m cells, where m is the total number of outputs of the combinational part of the core. It should be noted that the architecture in Figure 1 is just an application example and the proposed method is not limited to it.

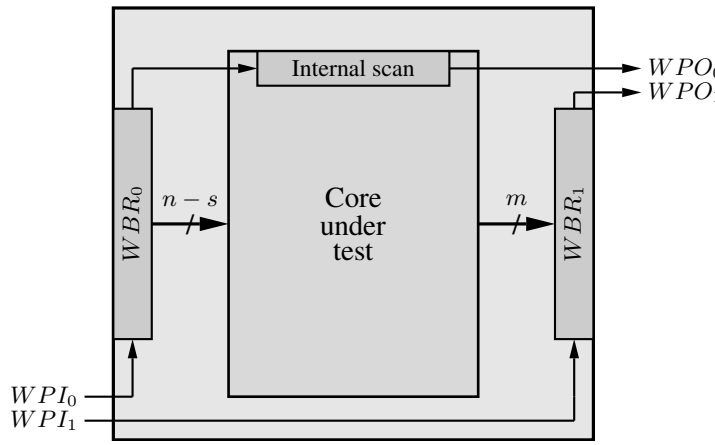


Fig. 1. Test application architecture

4 PROPOSED TEST GENERATION

Fault coverage and length are the two important test parameters. An ideal method based on genetic algorithm would track both of them and would try to evolve a test with the highest possible coverage and the shortest length. However, this would require to encode the whole test into the chromosome which would result in a search space of unacceptable size. The chromosome length would be variable which would make the length determination difficult and the population evaluation would be very slow.

SKEWGEN proposed in this paper maximizes the TDF coverage without tracking the test length. Therefore, the fitness of an individual is the number of detected TDFs after the possible acceptance of the individual. The test length is minimized only indirectly by assuming if the highest number of TDFs is detected in every step; then consequently fewer steps will be required for the same TDF coverage and the test will be shorter. This is a heuristic assumption of the developed method. SKEWGEN encodes only a part of the test (particularly one vector) into the chro-

mosome which significantly reduces the chromosome length. The test generation is divided into steps (in each step a local optimization as TDF coverage maximization is performed). Only one vector for PIs and PPIs is generated in each step instead of the vector pair usual for delay tests. The generated vector is appended to the test sequence generated in previous steps (the vector pairs, which will be applied to the logic core assuming the given test application architecture, are considered only during fault simulation and only in order to evaluate the generated vector — the result of test generation is the test sequence comprised of the generated vectors without any kind of overlap). The number of clock cycles required for the test application is the length of the test sequence.

The vector is used as the bit-string chromosome (i.e. the chromosome contains only one vector). Therefore, the chromosome length also equals to the number of PIs and PPIs of the logic core. Since the chromosome length varies from tens to thousands, SKEWGEN determines the population size as

$$g_1 + \left\lceil \frac{c}{g_2} \right\rceil,$$

where g_1 is the base population size, c is the chromosome length and g_2 is the divisor limiting the growth of the population size (g_1 and g_2 are user-defined parameters). The population will be g_1 for the smallest logic core and will increase with the problem size but will not become too large neither for the largest cores (larger problem size requires more starting points in the search for finding the global optimum instead of just some local one). For example, the problem size is 2^{10} for a logic core with 10 inputs and 2^{1000} for a core with 1000 inputs. SKEWGEN uses elitism to let one elite (the fittest individual) to unconditionally survive.

The evolution continues while the individuals improve. However, the improvement can be discontinuous and can proceed later. Therefore, population-threshold is used as the condition stopping the evolution, i.e. the number of populations in which the individuals do not improve. The threshold is scaled to save computation time in the early phase of test generation and to evaluate more populations later when the fault coverage grows slower. The developed population-threshold T_g used by SKEWGEN is given as

$$T_g = T_{\min} + \left\lceil T_{\max} \times \left(\frac{f_{\max}}{N_{TDF}} \right)^t \right\rceil,$$

where the minimum threshold $T_{\min}$, the maximum additional threshold $T_{\max}$ and the power of the scaling t are user-defined parameters; $f_{\max}$ is the fitness of the best individual in the current population (the number of detected TDFs) and N_{TDF} is the total number of TDFs. The population-threshold is shown in Figure 2 where the function T_g is illustrated for three different values of t. The bounds of T_g are given by $T_{\min}$ and $T_{\max}$ and the growth is defined by t. The value of T_g grows more slowly for $f_{\max}$ close to 0 and more rapidly for $f_{\max}$ close to N_{TDF} with higher values of t.

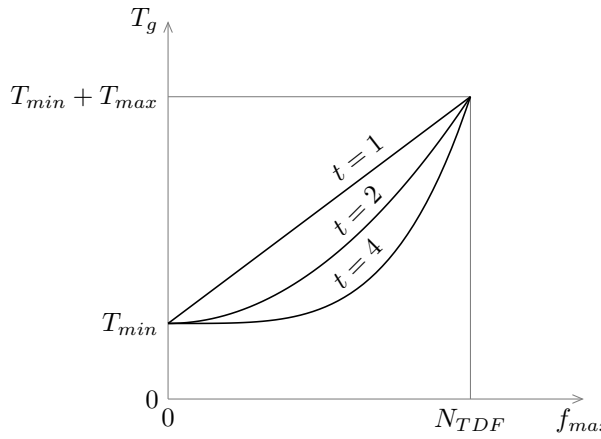


Fig. 2. Population-threshold scaling

4.1 Test Generation Flow

Figure 3 shows the flowchart of the proposed SKEWGEN. The test generation is executed as follows:

1. The population of individuals (vector candidates) is assembled randomly.
2. The fitness of each individual is determined by fault simulation.
3. New population is created repeatedly based on the previous population while $f_{\max}$ continuously improves.
4. If $f_{\max}$ ceases to improve and remaines constant in the previous T_g populations then any further evolution of the current population is discontinued.
5. If the acceptance of the best individual cannot improve the fault coverage then the test sequence is considered to be final and the test generation stops.
6. Otherwise, the best individual is accepted by appending the vector to the test sequence.
7. If there are more faults to be considered then the evolution of populations continues starting with a new random population.

Figure 4 shows an example of fitness development during test evolution, where $f_{\max}$ is the fitness of the best individual, $f_{\min}$ the fitness of the worst one, f_{avg} is the average of fitness in the population, the Roman numbers I, II, ..., VI mark the ends of test generation steps and letters A, B, C divide the third step into subparts in the magnified part of the graph. Every step of the test generation starts with a new random population and ends with the acceptance of the best individual (appending the vector represented by the chromosome to the test sequence). The ends of the steps are easily identifiable by a sudden significant change in the fitness.

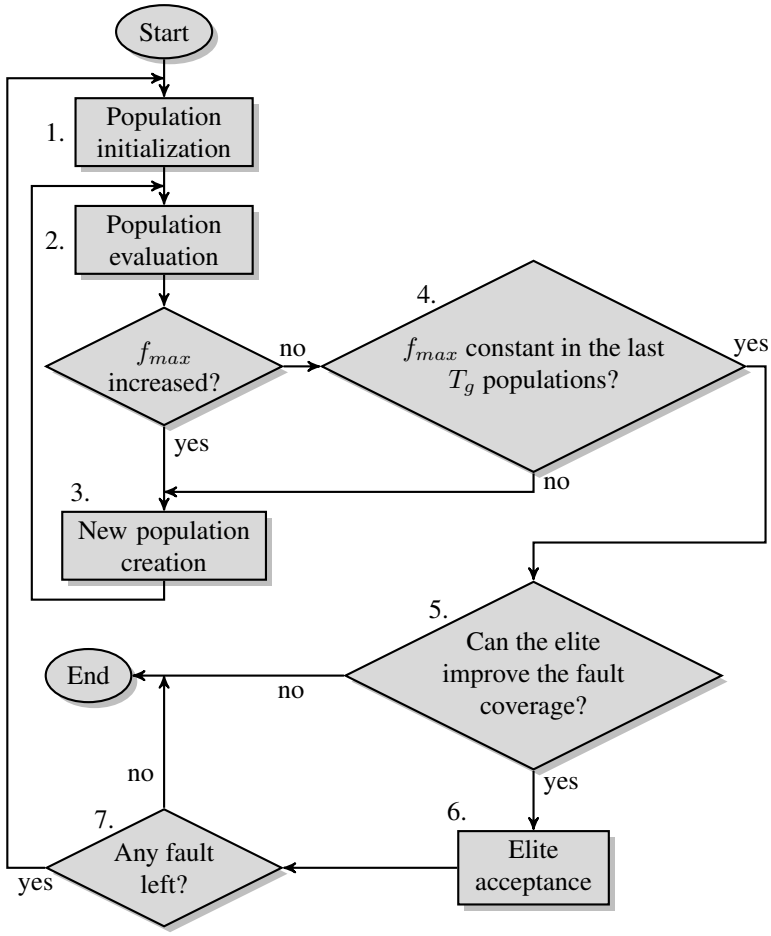


Fig. 3. Flowchart of the proposed test generator

For example, the fitness is approximately between 450 and 500 (450–500 detected TDFs) during the generation of the second vector but at the end of the step (marked as II) it changes and becomes more than 500. In that moment (marked as A) the test generation continues with a new random population. Immediately, the new population is significantly better than the previous one since the test generation is still in the early phase and detection of new faults is relatively easy. The population improves continuously since $f_{\max}$ increases. The improvement stops at B and remains the same until C. Between B and C exactly T_g populations are created and evaluated. The population still changes significantly as it is demonstrated by the changing tendency of f_{avg} and $f_{\min}$, but $f_{\max}$ remains the same because the best individual is protected by elitism. The generation of the third test vector is concluded at C

because the population was not improved in the previous T_g populations. The generation of the fourth vector follows with a new initial population. The evolution of other test vectors is very similar. The test generation in the example ends with the acceptance of the sixth vector marked as VI because the seventh vector (not shown in the figure) cannot improve the fault coverage.

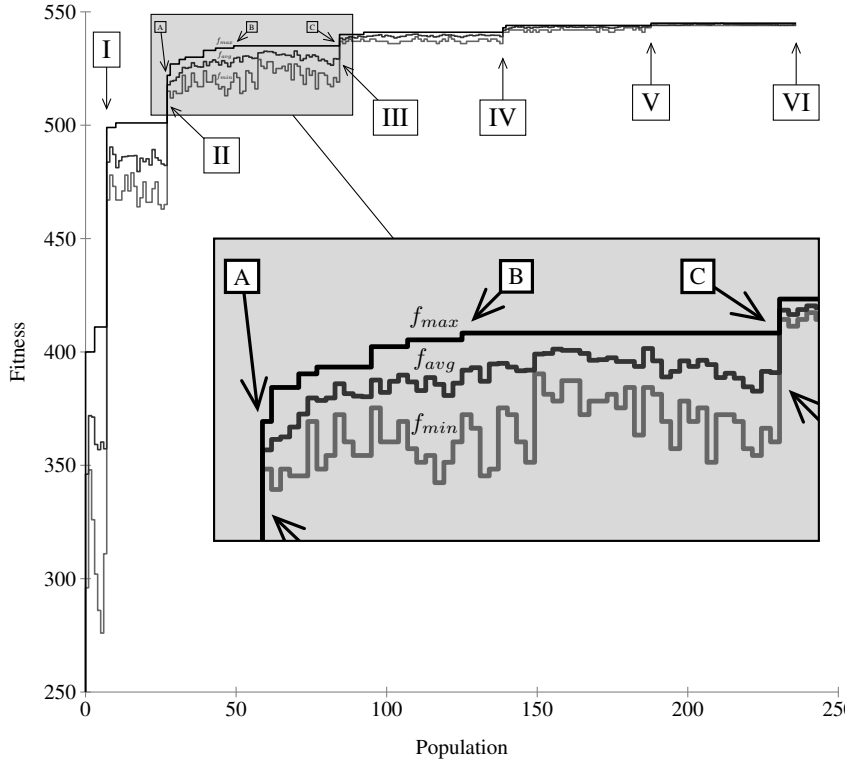


Fig. 4. Test evolution for s344 benchmark circuit

4.2 Solution Evaluation

The evaluation of the individuals of the population is executed by fault simulation. The individual is the test-vector candidate encoded directly by bit-string chromosome. The fault simulation determines the fitness, i.e. the TDF coverage which would be achieved by the acceptance of the individual. The vector pairs for fault simulation are assembled based on the test application scheme: the vector is sent into the WBR and the content of the WBR is applied in every clock cycle (therefore, an excitation vector is applied to the core in every cycle). The proposed SKEWGEN can be used with any other test application scheme by modifying the way the vector pairs are assembled for fault simulation.

An example of population evaluation is shown in Figure 5. The test sequence already contains vectors V_1 and V_2, and the development of V_3 is in progress. The current population of individuals is evaluated. Vector candidates `1010` and `0010` were already evaluated and fitness values 9 and 8 were determined. The evaluation of candidate `1100` is executed as follows. The test vector is a bitstream which will be shifted in to the WBR and scan chain. The bits in WBR and scan chain are continuously applied to the core. Therefore, each bit creates a new excitation vector for which the initialization vector is the previous excitation vector. Figure 6 shows how the vector pairs are created for fault simulation.

1. The initialization vector of vector pair P_1 is the vector V_2. The excitation vector is created by one-bit shift of the initialization vector and appending to it the first bit of vector V_3.
2. The initialization vector of vector pair P_2 is the excitation vector of P_1. The excitation vector of P_2 is created by a one-bit shift of this initialization vector, and appending to it the second bit of vector V_3.
3. The other vector pairs are generated similarly.

There are always as many vector pairs for fault simulation as bits in the vector (chromosome length). The construction of vector pairs always requires the previous vector as demonstrated by the example. A vector containing the reset values for WBR and scan cells is considered as the previous vector during the evolution of the first vector. The reset values are the assumed initial values (in the beginning of the test). These values can also be undefined if the reset values are unknown.

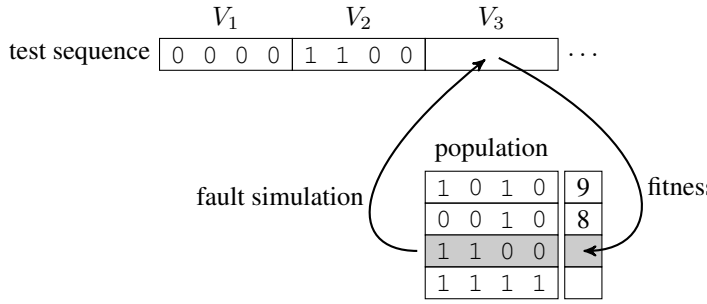


Fig. 5. Population evaluation

Definition 1. Let $U = (u_1, u_2, \ldots, u_o)$ be a vector with length o and $V = (v_1, v_2, \ldots, v_p)$ be a vector with length p, then $(U \# V) = (u_1, u_2, \ldots, u_o, v_1, v_2, \ldots, v_p)$ is the concatenation of vectors U and V.

Definition 2. Let $W = (w_1, w_2, \ldots, w_q)$ be a vector with length q, then $(W)_{[i,j]} = (w_i, w_{i+1}, \ldots, w_j)$ is the sub-sequence of W with indexes i and j, where $1 \leq i \leq j \leq q$.

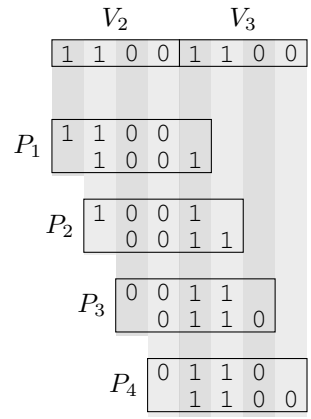


Fig. 6. Vector pairs for fault simulation

Definition 3. Let the vector V_0 contain the reset values of WBR and scan cells, then the sub-sequence $(V_{k-1} \# Z)_{[m,m+c-1]}$ is the initialization vector and the sub-sequence $(V_{k-1} \# Z)_{[m+1,m+c]}$ is the excitation vector of vector pair P_m for every $m \in \{1, 2, \ldots, c\}$ during the test generation of vector V_k, where Z is the vector candidate, c is the chromosome length and $k \geq 1$.

4.3 Population Evolution

The evolution is performed by means of genetic operators. Classical operators are applied since simple bit-string chromosome is used (these operators are described e.g. in [25]). The first new individual of the new population is always the fittest member of the previous population. The other individuals are created by selection, crossover and mutation. The fitness variance of the given search problem is low, and therefore rank selection is used to keep the diversity of the population. Single point crossover is executed over the selected two parents with a user-defined probability. The crossover is followed by bit-flipping mutation. The mutation probability can also be changed by the user.

5 PROPOSED METHOD FOR DETERMINISTIC INITIALIZATION OF POPULATIONS

Random initial population is sufficient in the early phase of test generation when the fault coverage grows relatively fast. However, usually the last faults are hard to detect as it is demonstrated by the following example. Let the length of the chromosome be 100 (which is only a very moderate value for the given problem); then the size of the search space is 2^{100} (considering two-valued logic). Let us assume that there is only one undetected fault for which only one test vector exists,

then the probability to find it is only 1 out of 2^{100}. Genetic algorithms perform guided search but in this case the other vectors will give no guide or information for finding the desired vector. Therefore, the vector can be found only by exhaustive or deterministic search and the success of the method with random initial population is almost beyond possibility.

The proposed SKEWGEN uses random population initialization only in the early phase of test generation. Deterministic initialization is used after a user-defined TDF coverage has been achieved. SKEWGEN uses two different mutation probabilities in these two initialization phases because the types of the searches differ significantly. These probabilities can be set by the user but it is recommended to use a high and low value during random and deterministic initialization, respectively. High mutation probability during random search ensures wider investigation of the search space, allows to achieve rapid TDF coverage growth and does not have negative impact because the best individual is always protected by elitism. Deterministic initialization is aimed to start the evolution with a population consisting of good individuals with the ability to detect another TDFs. High mutation probability during deterministic initialization would be counterproductive because it would destroy the good individuals since the elitism can not protect all of them. Therefore, very low mutation probability is recommended during deterministic initialization.

The principle of the developed deterministic population initialization is shown in Figure 7 where "init" and "excit" represent the circuit model for the initialization and excitation vector, respectively. For each individual of the population a different and previously undetected TDF is selected and modeled. The "init" model will ensure the initialization of the given circuit line to the value requested by the selected TDF. The "excit" model will detect the stuck-at fault on the same line as it is required by the TDF. These models are SAT-based models defined in [11]. The inputs of these adopted models are interconnected by the developed method to generate skewed-load test vectors (the inputs of the "excit" model are connected in a shifted way to the inputs of the "init" model). The interconnected inputs are labeled $I_0, \ldots, I_4$ in the figure.

- If the interconnected model is unsatisfiable then the fault cannot be detected for the given input order (WBR and scan cell order) and will not be targeted any more.
- Otherwise, the model is completed by requesting the last bit-value of the test sequence to be on the first intput of model "init". This request in the figure is the last bit of V_2, i.e. `0`, to the input I_0.
  - If the model is still satisfiable with this constraint then the values of the inputs determined by SAT solving are accepted as the individual. (The values of inputs $I_1, \ldots, I_4$ are assigned to the undefined values `XXXX` shown in the figure.)
  - Otherwise, another undetected TDF is selected and the process is repeated from the beginning.

- If there is no more TDF which previously was not targeted successfully and still more individuals are required then the population is filled up with random individuals.

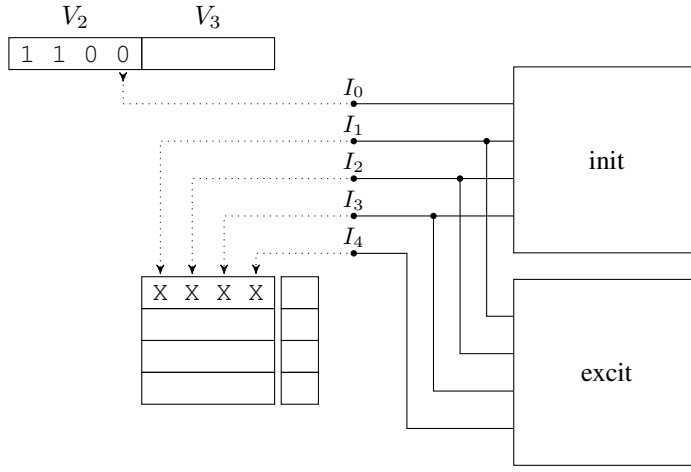


Fig. 7. Deterministic initialization

Definition 4. Let the vector V_0 contain the reset values of WBR and scan cells, the sub-sequence $(V_{k-1} \# Z)_{[c,2c-1]}$ be the initialization and Z the excitation vector for an undetected TDF then Z is a deterministic vector candidate for test vector V_k, where c is the chromosome length and $k \geq 1$.

It is very unlikely, but still a very special situation can occur when none of the remaining TDFs can be detected with the last bit of the test sequence on the first input of "init" model. This situation can be solved by appending the negated value of the last bit to the test sequence and shifting the boundaries of the last vector to include the new bit. This is the only situation when a bit and not a whole vector is added to the test sequence.

The output of the proposed deterministic population initialization is g_{det} vector candidates for g_{det} different TDFs which are detectable but still undetected, and $g - g_{det}$ random vector candidates, where g is the number of individuals in the population and $1 \leq g_{det} \leq g$.

Theorem 1. SKEWGEN with the proposed deterministic population initialization always achieves the highest possible TDF coverage for the given WBR and scan cell order.

Proof. Deterministic initialization is used and therefore the population always contains at least one vector for a detectable but until now undetected TDF assuming

the given WBR and scan cell order. Therefore, at least one vector will be in the new population which will increase the $f_{\max}$. There are two possible scenarios:

1. The proposed method will be able to further increase the $f_{\max}$ by creating a vector which is able to cover more undetected TDFs. The elitism will protect it and the further evolution could not destroy or modify it. If there will be further improvements then the elitism will grant protection similarly.
2. The proposed method will not be able to further increase the $f_{\max}$ but the elitism will protect the vector until the end of the population evolution.

   With regard to these two scenarios it is clear that every new vector will detect at least one previously undetected TDF. Since SKEWGEN will continue until the deterministic population initialization can provide at least one vector for at least one undetected TDF, eventually every TDF detectable for the given WBR and scan cell order will be covered.

□

6 RESULTS

The developed SKEWGEN was implemented in C++. Minisat 2.2.0 [12] was used as the SAT solver for deterministic population initialization. The evaluation was executed over ISCAS85, ISCAS89 and ITC99 benchmark circuits used as logic cores of SOCs and on a desktop computer (Intel Core2 Duo 2.8 GHz, 4 GB memory). The results were achieved assuming 90 % crossover probability; 50 % mutation probability during random initialization and 5 % during deterministic initialization; population-threshold scaling T_g with minimum threshold $T_{\min} = 2$, maximum additional threshold $T_{\max} = 50$ and scaling power $t = 4$; population size determination with base size $g_1 = 10$ and divisor $g_2 = 500$; and deterministic initialization performed from 80 % TDF coverage. These parameters were determined by experiments as the best values for the given search problem (they influence the test lengths only). The experimental results provided in this section are based on multiple runs (however, all the runs resulted in the same fault coverage, i.e. the achieved fitness values were identical, as the consequence of Theorem 1).

Table 1 shows the TDF coverage comparison to other recent methods for SOCs. The maximum achievable TDF coverage is influenced by the WBR and scan cell orders. Many of these orders were evaluated and the one with the best TDF coverage was selected. The results achieved for this order by the method published in [10] are shown in the "reord." column. The proposed SKEWGEN was evaluated with the same WBR and scan cell order for objective comparison, therefore the same TDF coverages are expected. The results in the "rand." column demonstrate that random population initialization is not sufficient to achieve full fault coverages (only for the three smallest circuits the coverages are identical to those achieved by the previous method). It is very difficult to detect some TDFs when the search space is too huge and random initial population is used. However, the same results (the

"deter." column) are achieved for every circuit when the proposed deterministic population initialization is used. It should be noted that the achieved results are the maximal achievable TDF coverages for the given WBR and scan cell order. This is also proved by the fact that both methods achieved the same TDF coverages assuming the same WBR and scan cell order. Each undetectable fault is due to the shift dependence of skewed-load tests and is not the limitation of the proposed SKEWGEN method.

Circuit	TDFs	[3]	[10]	reord.	SKEWGEN	
					rand.	deter.
c880	1 582	85.0	96.90	99.24	98.80	99.24
c1355	2 566	59.8	97.78	99.65	98.40	99.65
c1908	2 614	82.1	97.51	99.46	99.43	99.46
c3540	5 208	73.6	90.44	94.60	94.55	94.60
c5315	8 248	87.3	96.37	99.19	99.16	99.19
s344	552	92.2	94.20	98.73	98.73	98.73
s382	646	86.1	87.77	98.14	98.14	98.14
s526	948	85.6	87.87	92.09	92.09	92.09
s832	1 614	71.3	73.30	84.70	84.57	84.70
s1196	2 110	80.3	85.55	95.78	95.50	95.78
s1423	2 512	90.2	95.50	98.61	98.49	98.61
s5378	6 952	82.0	92.23	98.56	98.04	98.56

Table 1. TDF coverage comparison

The proposed SKEWGEN with the developed deterministic initialization achieves as high TDF coverages as the best current method. Furthermore, the "Reduc." column of Table 2 shows that at the same time the test is shorter in average by 25.75 % (in comparison to the "reord." column which contains the results for SKEWSAT with the same TDF coverage and assuming the best fault order among the evaluated ones resulting in the shortest test, since the test length for SKEWSAT is influenced by the order of targeting the faults [10]). The generated test was longer for circuit c1355 only probably because of the presence of hardly and just individually detectable faults. The test length reduction was possible by generating vectors covering always the highest number of (previously undetected) TDFs and in consequence this resulted in a less number of vectors and shorter tests. The tests are shorter even in the cases when compared to methods with lower TDF coverages. The proposed SKEWGEN has better TDF coverages in comparison to SfW method [3] but also shorter tests for each circuit except s832. In comparison to SKEWSAT [10] it has better TDF coverages in every case and still shorter tests for each circuit except c1355, c1908 and s832.

Circuit	[3]	[10]	reord.	SKEWGEN	Reduc. (%)
c880	7 523	4 128	2 224	781	64.88
c1355	17 565	1 124	1 092	1 968	−80.22
c1908	16 596	1 107	1 559	1 386	11.10
c3540	12 905	6 846	3 710	1 650	55.53
c5315	96 327	3 177	2 220	1 246	43.87
s344	614	198	177	120	32.20
s382	473	306	253	192	24.11
s526	914	1 285	844	576	31.75
s832	854	799	1 409	1 150	18.38
s1196	2 443	3 610	2 107	1 665	20.98
s1423	13 462	3 505	2 950	1 274	56.81
s5378	40 871	20 974	11 253	7 918	29.64
				Average:	25.75

Table 2. Test length comparison

The proposed SKEWGEN achieves not only as high TDF coverages as the best recent method and does that with shorter tests but has also better scalability. Table 3 shows additional results for the biggest available benchmark circuits which were too complex for the previous similar methods. The experiments show similar good fault coverages and relatively short test lengths. The worst coverage is 88.86 % for circuit s35932 where the shift dependence could not be resolved better by WBR and scan cell reordering. For every other circuit the achieved coverages are significantly higher.

The test generation times are shown in Figure 8 as function of the total number of TDFs because the most time consuming part of the developed method is the fitness evaluation performed by fault simulation, which depends mainly on the circuit size. The test generation time is almost negligible for circuits under 15 000 TDFs, it took less than 20 hours for most of the benchmark circuits, except b22s (42 hours), s38417 (67 hours) and b17s (122 hours). The achieved results show exponential growth of the test generation time, but even the longest test generations are worth the time because good fault coverages were achieved and the test length reduction could save much more expensive test application time. The TPGs without test compaction are usually faster but the consequent compaction still takes time because there is a large amount of possibilities for overlapping the generated vector pairs ($n!$, where n is the number of test vector pairs). The test generation of the proposed SKEWGEN takes hours and, exceptionally, even days, while the final test sequence can be used immediately to test the manufactured products, and the saved test cost is significant thanks to the shortness of the tests. Another disadvantage of the TPGs without

Circuit	TDFs	TDF cov. (%)	Length (bit)	Vectors	Populations
c6288	12 478	99.54	128	4	176
c7552	11 300	98.80	22 356	108	6 500
s9234	11 324	93.55	56 810	230	11 899
s13207	15 584	98.30	73 500	105	8 975
s15850	18 998	96.69	111 205	182	12 639
s35932	63 502	88.86	3 526	2	70
s38417	49 708	99.62	367 744	221	19 148
s38584	60 954	96.70	149 328	102	8 010
b14s	22 050	97.88	134 068	484	25 187
b15s	41 564	97.83	174 115	359	27 671
b17s	115 286	97.94	949 608	654	52 508
b20s	43 688	98.03	399 852	766	44 059
b21s	45 648	98.19	381 582	731	43 637
b22s	69 640	98.21	829 894	1 082	63 126

Table 3. Results for additional benchmark circuits

compaction is that they usually provide a set of vector pairs where only one vector pair is provided for a given TDF. However, other vector pairs can exist for that TDF and it is not guaranteed that the generated vector pair can be overlapped the best way with the other pairs.

Figure 9 shows the average number of populations required to generate a test vector as the function of the chromosome length (number of PIs and PPIs). The results demonstrate that the developed SKEWGEN does not perform exhaustive search. The average population number stays below 90 even with increasing size of the search space. The test quality (TDF coverage, test length) remains good for various problem sizes because the proposed scaling of the population size ensures adequate number of starting points for the search (therefore, the global optimum can be found more confidently even for larger logic cores). Alternatively, the number of populations could be scaled instead of the population size, but that would ensure more accurate local optimums only.

Objective comparison of the developed SKEWGEN to methods not considering wrapper-based SOCs is not possible. The PIs of sequential circuits not embedded in SOCs are accessible from the outside and shift-in through WBR is not necessary. This fact has two consequences: (1) The shift dependence of skewed-load tests is lower, and therefore higher fault coverages can be achieved. (2) Values on the PIs can be changed at any time without shift-in through WBR, and therefore the test is expected to be shorter. However, the developed SKEWGEN performs well even

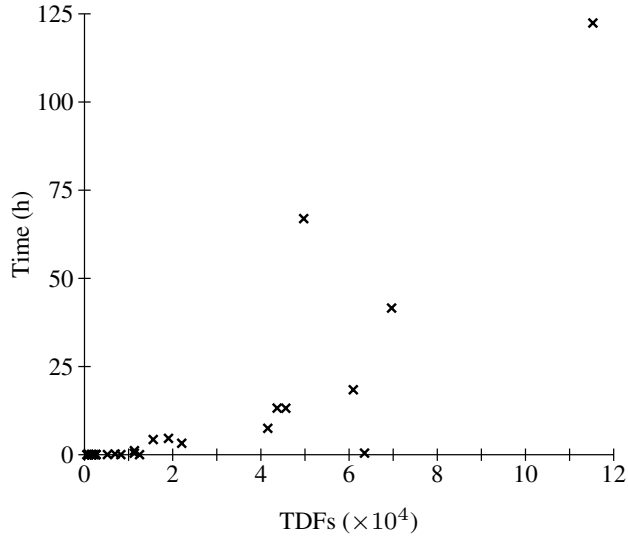


Fig. 8. Test generation time

in comparison to these methods. Table 4 shows TDF coverage comparison to the most recent methods.

SKEWGEN has worse coverages for small circuits in comparison to method [18] but starting with s5378 the achieved results are better (6 better results out of 10), and also better coverages were achieved for 9 circuits out of 14 in comparison to [19].

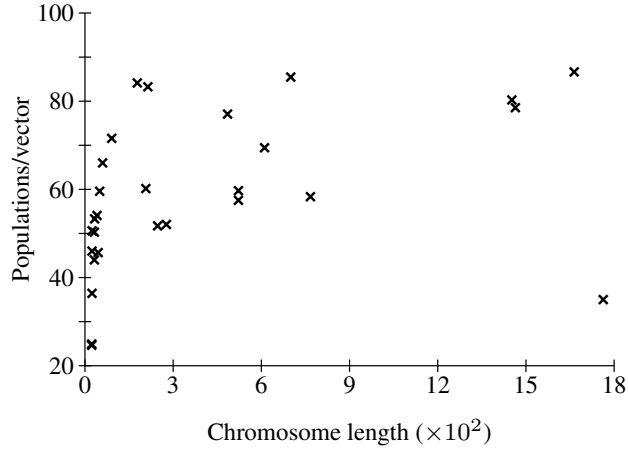


Fig. 9. Average number of populations

Circuit	TDFs	SKEWGEN	[18]	[19]
s344	552	98.73	-	97.97
s382	646	98.14	-	93.98
s526	948	92.09	-	93.35
s820	1 574	85.13	93.11	93.11
s953	1 738	95.86	98.79	98.79
s1196	2 110	95.78	100.00	100.00
s1423	2 512	98.61	98.66	98.66
s5378	6 952	98.56	97.86	97.86
s9234	11 324	93.55	93.35	93.24
s13207	15 584	98.30	96.83	96.80
s38417	49 708	99.62	99.54	-
b14s	22 050	97.88	95.02	85.75
b15s	41 564	97.83	-	92.16
b20s	43 688	98.03	95.77	90.86
b21s	45 648	98.19	-	84.25

Table 4. TDF coverage comparison to non-SOC methods

7 CONCLUSION

In this paper, SKEWGEN, the compressed skewed-load TPG based on genetic algorithm for wrapper-based logic cores of SOCs was proposed. The test application requires only one storage element per cell in scan chain and WBR, therefore it provides an appropriate way to test delay faults in low-overhead SOCs.

The developed method generates test vectors containing already overlapped test vector pairs. Consequently, less vectors are necessary to achieve the same TDF coverage and the test application will be shorter. The individuals of the initial new population are generated by SAT-based deterministic test generator. The individuals are created by targeting different TDFs. The task of the developed method is to evolve a vector which will detect not only one of these faults but as many as possible. The assembled circuit model for the given TDF allows to identify undetectable faults by classifying the problem as unsatisfiable. The new population contains always at least one vector which is able to increase the TDF coverage. Therefore, SKEWGEN guarantees the highest achievable TDF coverage for the given WBR and scan cell order.

The proposed SKEWGEN was evaluated over ISCAS85, ISCAS89 and ITC99 benchmark circuits used as logic cores of SOCs. The experimental results show TDF coverages as high as those of the recent methods and, furthermore, the tests are shorter by approximately 25 %. SKEWGEN has also better scalability. The

test generation was previously not possible for the biggest available benchmark circuits.

The most time consuming part of SKEWGEN is the determination of the fitness values, therefore parallel evaluation could accelerate the test generation significantly. Another possibility is to reduce the chromosome lengths by using partial vectors and generating tests sequentially for sub-parts of the core. These modifications could be used without the necessity to modify the proposed method.

This work has been supported by Slovak national project VEGA 2/0034/12.

REFERENCES

[1] 1500-2005: IEEE standard testability method for embedded core-based integrated circuits. Available on: http://dx.doi.org/10.1109/IEEESTD.2005.96465, 2005.

[2] International technology roadmap for semiconductors, 2010 update. Available on: http://www.itrs.net/Links/2010ITRS/Home2010.htm.

[3] Baláž, M.: SfW Method: Delay Test Generation for Simple Chain Wrapper Architecture. In: 28th Norchip Conference, 2010, p. 4.

[4] Baláž, M.—Dobai, R.—Gramatová, E.: Delay Faults Testing. In: R. Ubar, J. Raik, and H. T. Vierhaus (Eds.): Design and Test Technology for Dependable Systems-on-Chip, 2011, pp. 377–394.

[5] Balcárek, J.—Fišer, P.—Schmidt, J.: Test Patterns Compression Technique Based on a Dedicated SAT-Based ATPG. In: 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD), 2010, pp. 805–808.

[6] Balcárek, J.—Fišer, P.—Schmidt, J.: Techniques for SAT-Based Constrained Test Pattern Generation. In: 14th Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD), 2011, pp. 360–366.

[7] Bushnell, M. L.—Agrawal, V. D.: Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits. Kluwer Academic Publishers, 2000.

[8] Chen, Z.—Xiang, D.: A Novel Test Application Scheme for High Transition Fault Coverage and low Test Cost. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 29, 2010, No. 6, pp. 966–976.

[9] Cheng, K. T.—Devadas, S.—Keutzer, K.: Delay-Fault Test-Generation and Synthesis for Testability Under a Standard Scan Design Methodology. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 12, 1993, No. 8, pp. 1217–1231.

[10] Dobai, R.—Baláž, M.: SAT-Based Generation of Compressed Skewed-Load Tests for Transition Delay Faults. In: 14th Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD), 2011, pp. 191–196.

[11] Drechsler, R.—Eggersglüss, S.—Fey, G.—Tille, D.: Test Pattern Generation using Boolean Proof Engines. Springer, 2009.

[12] Eén, N.—Sörensson, N.: An Extensible SAT-Solver. In: Theory and Applications of Satisfiability Testing, Lecture Notes in Computer Science, Springer, 2004, pp. 502–518.

[13] Eggersglüss, S.—Fey, G.—Glowatz, A.—Hapke, F.—Schloeffel, J.—Drechsler, R.: MONSOON: SAT-Based ATPG for Path Delay Faults Using Multiple-Valued Logics. Journal of Electronic Testing, Vol. 26, 2010, No. 3, pp. 307–322.

[14] Eggersglüss, S.—Tille, D.—Fey, G.—Drechsler, R.—Glowatz, A.—Hapke, F.—Schloeffel, J.: Experimental Studies on SAT-Based ATPG for Gate Delay Faults. In: 37th International Symposium on Multiple-Valued Logic, 2007, p. 6.

[15] Mao, W.—Ciletti, M. D.: Reducing Correlation to Improve Coverage of Delay Faults in Scan-Path Design. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 13, 1994, No. 5, pp. 638–646.

[16] Namba, K.—Ito, H.: Chiba Scan Delay Fault Testing with Short Test Application Time. Journal of Electronic Testing, Vol. 26, 2010, No. 6, pp. 667–677.

[17] Pomeranz, I.: Concatenation of Functional Test Subsequences for Improved Fault Coverage and Reduced Test Length. IEEE Transactions on Computers, Vol. 61, 2012, No. 6, pp. 899–904.

[18] Pomeranz, I.: On the Computation of Common Test Data for Broadside and Skewed-Load Tests. IEEE Transactions on Computers, Vol. 61, 2012, No. 4, pp. 578–583.

[19] Pomeranz, I.: Static Test Compaction for Delay Fault Test Sets Consisting of Broadside and Skewed-Load Tests. In: 29th VLSI Test Symposium (VTS), 2011, pp. 84–89.

[20] Pomeranz, I.—Reddy, S. M.: Fixed-State Tests for Delay Faults in Scan Designs. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 19, 2009, No. 1, pp. 142–146.

[21] Pomeranz, I.—Reddy, S. M.: Functional and Partially-Functional Skewed-Load Tests. In: 15th Asia and South Pacific Design Automation Conference (ASP-DAC), 2010, pp. 505–510.

[22] Savir, J.: Skewed-Load Transition Test: Part I, Calculus. In: International Test Conference, 1992, pp. 705–713.

[23] Savir, J.—Patil, S.: Scan-Based Transition Test. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 12, 1993, No. 8, pp. 1232–1241.

[24] Savir, J.—Patil, S.: Broad-Side Delay Test. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 13, 1994, No. 8, pp. 1057–1064.

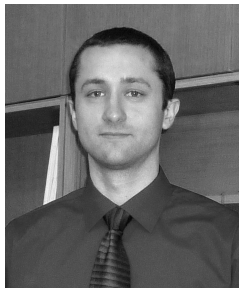
[25] Sivanandam, S. N.—Deepa, S. N.: Introduction to Genetic Algorithms. Springer, 2008.

[26] Stareček, L.—Sekanina, L.—Kotásek, Z.: Reduction of Test Vectors Volume by Means of Gate-Level Reconfiguration. In: 11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems, 2008, pp. 255–258.

[27] Šimáček, J.—Sekanina, L.—Stareček, L.: Evolutionary Design of Reconfiguration Strategies to Reduce the Test Application Time. In: Evolvable Systems:

From Biology to Hardware, Lecture Notes in Computer Science, Springer, 2010, pp. 214–225.

[28] Wang, S.-J.—Peng, K.-L.—Hsiao, K.-C.—Li, K. S.-M.: Layout-Aware Scan Chain Reorder for Launch-Off-Shift Transition Test Coverage. ACM Transactions on Design Automation of Electronic Systems, Vol. 13, 2008, No. 4, Art. No. 64.



Roland Dobai received the B. Sc. and M. Sc. degrees in computer engineering and the Ph. D. degree in applied informatics from the Slovak University of Technology in 2006, 2008 and 2011, respectively. Since 2008 he is with the Institute of Informatics of the Slovak Academy of Sciences in Bratislava (Slovakia). His research is targeted to test generation, fault simulation and built-in self-test of digital VLSI circuits.

Marcel Baláž received the M. Sc. degree in computer engineering and the Ph. D. degree in applied informatics from the Slovak University of Technology in 2003 and 2010, respectively. Since 2003 he is with the Institute of Informatics of the Slovak Academy of Sciences in Bratislava (Slovakia). His research is targeted to test architectures, built-in self-test, test generation of digital cores for Systems on Chip.