Testability Improvements Based on the Combination of Analytical and Evolutionary Approaches at RT Level

Josef Strnadel, Zdeněk Kotásek
Faculty of Information Technology, Brno University of Technology
Božetěchova 2, 612 66 Brno, Czech Republic
{strnadel, kotasek}@fit.vutbr.cz

Abstract

In the paper a new heuristic approach to the RTL testability analysis is presented. It is shown how the values of controllability/observability factors reflecting the structure of the circuit and other factors can be utilised to find solutions which are sub-optimal but still acceptable for the designer. The goal of the methodology is to enable the identification of such testability solutions which satisfy concrete requirements in terms of the number of registers included into the scan chain, the area overhead and the test application time as a result of RTL testability analysis. The approach is based on the combination of analytical and evolutionary approaches at the RT level.

1. Introduction

In the past, numerous methodologies based on enumerating controllability/observability factors were published. These factors are utilised in different heuristic approaches, the identification of registers through which the test will be applied is the result of implementing these methodologies. During the selection, the requirements of a designer or a test expert must be taken into account and satisfied, if possible. This process can be realised as a strictly heuristic approach in which different modifications are accepted and evaluated successively, the solution which satisfies the requirements in the best way is then selected to be implemented. We see that these procedures might be quite time consuming in certain situations and for certain RTL circuit structures, e. g. those with numerous feedback loops. Moreover, we feel that controllability/observability factors are not the only factors which should be included into the "making decision process" the result of which is the selection of registers for *scan*.

Many approaches to the RTL *testability analysis* were demonstrated so far. Most of them are based on the *UUA* (Unit Under Analysis) structural analysis. Recently, the methodologies whose goal is to analyse VHDL design description and insert scan at the behavioural level were presented [1]. The methodology is based on locating memory elements in the circuit. Then, all located memory elements are inserted in the *scan chain* (see Figure 1) at the behavioural level. The approach is supposed to be generalised to cover multiple scan chains.

Scan approaches fall into two main groups: *full scan* and *partial scan*. In partial scan, flip-flops (FFs) are selected in such a way that the remainder of the circuit has certain desirable testability properties. Existing approaches for selecting FFs for partial scan can be classified as *testability analysis based* [2], *test generation based* [3] and *structural analysis based* [4]. All of these techniques suggest testability modifications after the completion of the design and are incapable of suggesting behavioural modifications by identifying testability bottlenecks in the design behaviour during the design process. Some methods exist which are based on inserting test registers in order to obtain self-testable circuits [5], [6]. The methodologies are implemented in the way which guarantees minimum hardware overhead [7]. In [8] an interesting method how to further utilise scan chains is demonstrated - they are used for pattern decompression.

Several algorithms were developed to select partial scan FFs to break feedback cycles [9][10]. In [11] a methodology is described, which is able to take into account a number of technological constraints and determines the optimal order of FFs into the scan chains.

In our paper we describe the methodology which utilises analytical and evolutionary approaches to identify an acceptable solutions to the problem of selecting FFs for the scan. The input to the methodology (see Figure 7) is the circuit structure, the database which contains the information of the impact of utilising certain decisions (in terms of area overhead and test application time etc.).

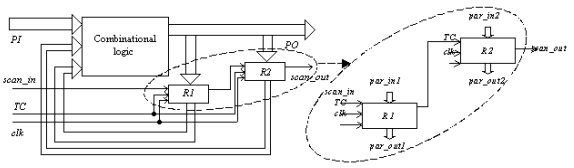


Figure 1. Scan chain example and its simplified schema

2. Definition of the Problem and Basic Concepts

In design for testability methodologies different aspects (criteria) are taken into account which allow different solutions of the problem to be gained. As a result of such approaches, a subset of registers is identified through which the test will be applied. It is supposed that the registers will be converted either to *scan registers* or *BIST elements*.

One possibility how to identify the set of registers for the test application process is by means of two steps: 1) enumerating all combinations of registers which could possibly form the scan chain(s) and 2) evaluating every alternative. These approaches are usually denoted as "rough methods" (although they lead to acceptable solutions, they can be too much time consuming for the problem complexity). These methods are based on *exhaustive search*: they simply visit all points in the search space in some order and retain the best solution visited. Other methods only visit part of the search space, albeit the number of points visited may grow exponentially (or worse) with the problem size. To avoid this problem, it is possible to use general-purpose heuristics that do not guarantee an optimal solution.

In this paper it is presented how the combination of analytical and evolutionary approaches can be used for the RTL testability analysis. In our methodology, the evolutionary approach is represented by genetic algorithm which performs a multidirectional search by maintaining a *population* of potential solutions (set of registers for scan). The population of registers for scan undergoes a simulated evolution from one generation to another: at each generation the relatively good solutions reproduce, while the relatively bad solutions die. The goodness or badness of the solutions is defined by a cost function. Each solution is encoded as a *chromosome which* is represented as a string of bits from a binary alphabet. To generate new solutions, *crossover* operation is used. For the crossover, two solutions S_1 and S_2 of the current generation of registers are selected and the chromosome to the new solution is produced. The new chromosome is the result of mixing a part of the chromosome of S_1 with a part of that corresponding to S_2. This means that the new solution inherits certain features of its two parent solutions. The *mutation* operator, on the other side, produces a small, random perturbation to a given solution (chromosome).

To determine the solutions which are considered as parents for crossover operations, *selection scheme* is used. An essential criterion at selection for crossover is the *fitness* of the solution, defined by the cost function. Thus, selection of fit solutions for crossover ensures the propagation of high quality features into the next generation. The selection is based on a probabilistic scheme which favours candidates having a high fitness. Producing several successive generations, the average fitness of the solutions is increasing. The algorithm is usually stopped after a certain number of iterations or when no further improvements are produced. The best solution that has been produced is one which is hopefully close to the optimum. To apply genetic algorithm to a problem, it is necessary to identify: (1) meaningful representation for the candidate solutions; (2) a fitness function to assess different solutions; and (3) a set of useful genetic operators, that can efficiently recombine and mutate candidate solutions.

An optimisation problem is mapped into the problem of finding the most-fit *individual* within a population during an evolution process. Fitness is measured by a fitness function which is related to the objective function of the optimisation problem.

We did the research in two stages – during the first one we analysed the number of all unique solutions (i. e. the size of a state-space) and during the second one we dealt with a design and implementation of an effective methodology based on evolutionary approach. The goal of the methodology is the identification of sub-optimal or near optimal solutions (i. e. those with an acceptable trade-off between user requirements and circuit structure properties), the search for these solutions is done in the reduced state space without the need to explore the complete state-space containing all possible solutions.

The paper is organised as follows. First, the methodology is described, then the methodology based on genetic algorithm together with experimental results is presented. Finally, the perspectives of future activities in this research area are discussed.

3. The Description of the Methodology

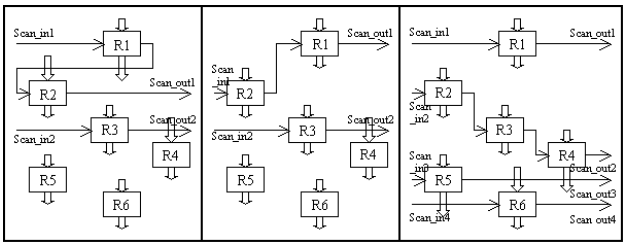
For the purposes of the methodology we denoted the set of all registers in UUA as $REGS_{UUA}=\{R_1, R_2, \ldots R_n\}$. The basic idea of the first stage of our research is as follows: It is true that numerous alternatives how to implement a scan structure in a UUA exist if e. g. each of these structures can consist of several parallel scan chains in

which the order of registers 1) is important for the result (alternative (1)), 2) is not important for the result (alternative (2)). It is naturally supposed that the goals of the analysis in terms of area and pin overhead and testability properties are defined before the analysis starts. In this paper, we shall deal primarily with the alternative according to 1) which is more general. The alternative 2) can be solved after investigating the alternatives according to 1).

As the first step of our analysis, we calculate how big the partial state space is, i. e. how many scan structures exist, each of them consisting of k scan registers exactly, $\{r_1, r_2,\ldots, r_k\} \subseteq REGS_{UUA}$, where $1 \le k \le n$. First, we present the partial state-space analysis (for given k), then the complete state-space analysis (for general n) for both alternatives (1) and (2) will be presented.

For the methodology we introduced the following notation in which:

- the ordered (1) or non-ordered (2) sequence of scan registers will represent scan registers belonging to the same scan chain within this structure regarding the order (1) or regardless of the order (2),
- the period character ('.') will separate parallel chains configured in the UUA,
- every $r \in \{r_1, r_2,\ldots, r_k\}$, will be involved exactly in one scan chain within the resulting scan structure.



$R_1R_2.R_3$ (k=3, n=6) $R_2R_1.R_3$ (k=3, n=6) $R_1.R_2R_3R_4.R_5.R_6$ (k=6, n=6)

Figure 2. Example of the relation between the notation and the structure

An example for n=6: let k=3, the scan structure consists of $R_1R_2.R_3$ registers (see Figure 2 a)) configured into two parallel ordered scan chains with R_1, R_2 (one scan chain), and R_3 (the other scan chain) registers; R_4, R_5 and R_6 registers are not modified into scan registers and thus not included into scan chains. Similarly, $R_2R_1.R_3$ notation (see Figure 2 b)) represents the structure containing two parallel ordered scan chains R_2, R_1 (one scan chain) and R_3 (the other scan chain) registers; R_4, R_5 and R_6 registers are not modified into scan registers and not included into scan chains. Let for k=6 the scan structure be $R_1.R_2R_3R_4.R_5.R_6$ (see Figure 2 c)), then the structure of four parallel scan chains is as follows: first scan chain - R_1, second scan chain - R_2, R_3, R_4, third scan chain – R_5, fourth scan chain – R_6. It is evident that if the order of registers is not taken into account (in case of alternative (2)), then $R_1R_2.R_3$ and $R_2R_1.R_3$ solutions are identical.

It is seen that by means of the above described notation any scan structure containing any combination of registers in scan chains (including their order in (1) case) can be identified uniquely.

From the mathematical point of view, the notation represents $\{r_1, r_2,\ldots, r_k\}$ set partition into j partition classes (called also blocks), where $1 \le j \le k$ (j is the number of concatenation characters in the notation increased by 1), and (in (1) case) the ordering of elements within each block is required, rather than the ordering of blocks. Let it be noted that a set partition of the set $\{r_1, r_2,\ldots, r_k\}$ is a collection $B_0,B_1,...,B_j$ of disjoint subsets of $\{r_1, r_2,\ldots, r_k\}$, their union is defined as $\{r_1, r_2,\ldots, r_k\}$ and each B_i is called a block.

Our partial task is defined as "how many different solutions of scan configuration based on including just k registers into the scan structure exist, $\{r_1, r_2,\ldots, r_k\} \subseteq REGS_{UUA}$, where $1 \le k \le n$?"; it means that we must reveal the number of partitions of the $\{r_1, r_2,\ldots, r_k\}$ set, when the order of the elements in the blocks is taken into account. Let the number of partitions for given k be denoted as $nparts_k$ which represents the partial state-space size**.** Note: if the order of registers in the scan chain is not taken into account (alternative (2)) then the order of registers according to the notation introduced will not be seen as ordered and then $nparts_k = b_k$, where b_k is a "Bell number" [12] (see Figure 3) determining the number of partitions of a set containing k elements.

$$b_k = \sum_{i=0}^{k-1}\left[\binom{k-1}{i} \times b_i\right], \qquad \text{where } b_0 = 1$$

Figure 3. Bell number recurrence relation

If we are able to determine the value of $nparts_k$ for given k, i. e. we are able to say how many unique scan structures exist in which every $r \in \{r_1, r_2,\ldots, r_k\}$ will be included into scan just once, then we are also able to determine the complete state-space size, i. e. how many different scan structures can be configured for general n if we create step by step all scan structures containing just 1, 2, up to exactly n registers from $REGS_{UUA}$ set (all 1, 2, ….n element subsets of $REGS_{UUA}$ set are formed and such scan structures are selected for each subset which contains every element from the subset just once).

For general n the number of scan structures will be equal to $nstructs_n$, for which we developed a formula:

$$nstructs_n = \sum_{k=1}^{n}\left[\binom{n}{k} \times nparts_k\right]$$

Figure 4. Formula for $nstructs_n$ evaluation

where the value of $\binom{n}{k} \times nparts_k$ represents the product of two values: the number of k-element subsets (in this case $\{r_1, r_2, \ldots, r_k\}$) of an n-element set (here $REGS_{UUA}$) and the number of different structures which can be formed from k registers $r_1, r_2, \ldots, r_k$, every $r \in \{r_1, r_2, \ldots, r_k\}$ will be involved in each scan structure just once.

While the problem of $nstructs_n$ evaluation for alternative denoted as (2) is solved because $nparts_k$ value for given k is known ($nparts_k = b_k$, see Figure 3), the problem of $nstructs_n$ evaluation for alternative denoted as (1) is more complicated because the order of the elements in the blocks must be taken into account, which means that the formula for $nparts_k$ (see Figure 5) evaluation is more complicated for the (1) alternative than for that one denoted as (2). For every block we must now take into account all possible permutations, which were not considered in the alternative (2) case. For this purpose it is necessary to determine $nparts_k$ evaluation formula for alternative (1) as well. The procedure of deriving the formula is quite complicated, therefore just the resulting formula is given here as:

$$nparts_k = \sum_{i=1}^{k}\left[\frac{k!}{i!} \times \binom{k-1}{i-1}\right]$$

Figure 5. Formula for $nparts_k$ evaluation (for alternative (1))

Let it be noted that the value $\frac{k!}{i!} \times \binom{k-1}{i-1}$ represents the number of ways in which a k-element set can be partitioned into i blocks, the order of elements in the blocks is taken into account. This value is often denoted as „Lah number" – for more detail information see [13] or [14].

For better understanding the size of the complete state-space consisting of $nstructs_n$ unique structures (solutions), the formula (Figure 4) for evaluating $nstructs_n$ for given n is completed for both alternative (1) and alternative (2) with tables (see Table 1 and Table 2) and a diagram (see Figure 6). The headings of the tables are described later in this paper.

Let it be noted that each of $nstructs_n$ solutions represents a structure which has various quality level in general. We have already solved the partial and complete state-space size problem presenting formula for $nstructs_n$ (Figure 4). The most important problem to be solved is still the problem of selecting the most appropriate solution of the scan organisation (i. e. selecting the best alternative from the state space) whereby the test application quality and price are taken into account.

The symbols in the rows of the Table 1 have the following meanings: 1) n - the number of registers in *UUA*, 2) $nstructs_n$ - the total number of unique scan structures which can be formed from n registers (alternative (1) case), 3) t_{all} - time needed to identify the best solution searching complete state-space, 4) t_{gen} - time needed to identify the best solution by means of our methodology based on genetic algorithm.

Table 1. State space size for alternative (1)

n	1	2	3	4	5	6
$nstructs_n$	1	5	25	147	1031	8463
t_{all}	1s	5s	25s	2min	17min	2,3h
t_{gen}	5s	10s	25s	90s	3min	7min

Similarly, in the Table 2 the symbols have following meanings: 1) n - the number of registers in *UUA*, 2) $nstructs_n$ - the total number of unique scan structures which can be formed from n registers (alternative (2)), 3) t_{all} - time needed to identify the best solution by searching through the complete state-space, 4) t_{gen} - time needed to identify the best solution by means of our methodology utilising a genetic algorithm.

Table 2. State space size for alternative (2)

n	1	2	3	4	5	6
$nstructs_n$	1	4	14	51	202	876
t_{all}	1s	4s	14s	51s	3min	15min
t_{gen}	5s	9s	17s	35s	110s	160s

The times written in rows 3) and 4) of the Table 1 and Table 2 are valid for PC with PentiumII/333MHz processor and 64MB RAM.

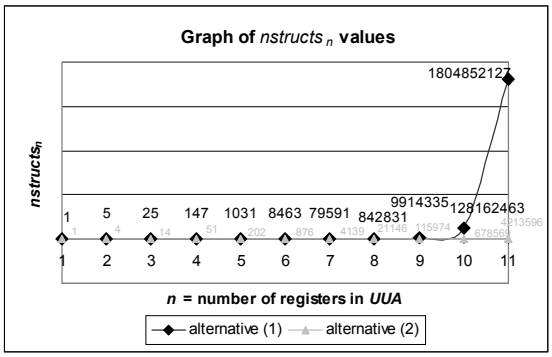


Figure 6. Comparative graph of $nstructs_n$ values for both alternatives

4. Testability Improvement Methodology Based on Evolutionary Approach

Main goals of the first stage of our research activities were presented in the previous section. During the second stage of our research we dealt with a design and implementation of an effective method based on evolutionary approach. The method is used to find sub-optimal or near optimal solutions (i. e. those with the best trade-off between user requirements) in the state space without the need to explore the complete state space. The main principles of this method are briefly presented in this section.

It can be seen (in row t_{all} of both tables Table 1 and Table 2) that a "rough method" used to find a solution (i. e. that with a best trade-off) from all $nstructs_n$ possible solutions (each representing a set of mutually parallel scan chains to be configured in *UUA*) is absolutely unacceptable for n>>1, because of the time complexity. In general, the time complexity can be decreased either by means of heuristics or by means of optimising techniques. For this purpose we decided to utilise an optimising genetic algorithm which allows to describe the problem by means of a binary string. Here, the optimisation can be seen in decreasing of a time needed for optimal solution finding significantly. The advantage of the approach can be primarily seen in the fast convergence of genetic algorithms to the searched solution.

To be able to utilise a genetic algorithm for our purpose, it was necessary to represent the problem by means of a binary string, denoted as chromosome. Let it be reminded that for *UUA* with n registers, $nstructs_n$ solution alternatives exist how to create various scan chain(s) configurations (i. e. possible unique solutions). Besides, it is required any alternative to be addressable by means of the chromosome. The chromosome $ch \in \{0,1\}^L$ is defined as a bit string with the length $L = \lceil log_2\ nstructs_n \rceil$. Then a chromosome can be seen as a unique address of a possible solution; there are $nstructs_n$ possible addresses, the address can acquire a value from <0; $nstructs_n$-1> interval. It means that concrete chromosome represents concrete solution from the complete state-space. A chromosome can be also seen as unique identification of 1) mutually parallel scan chains that will be inserted into a *UUA*, 2) a set of *UUA* registers that will be modified into scan registers and included into scan chains and 3) an order of scan registers in scan chain (in the alternative (1) case).

There are $nstructs_n$ distinct chromosomes and a state-space consisting of $nstructs_n$ unique solutions (each solution represents the set of scan chains which possibly configured in *UUA*), so it can be proved that a bijection from the chromosome set to state-space exists. Chromosome can be also seen as a prescription how to modify *UUA* to fulfil user requirements maximally (here using a scan technique) – with a certain trade-off penalty. The prescription determines the registers, which will be modified into scan registers, which scan chains will be configured in *UUA* and which scan registers will be involved into particular scan chains.

It is also necessary to evaluate which chromosome represents such *UUA* transformation offering the best diagnostic properties (i. e. the scan configuration with the best trade-off according to user requirements). For these purpose, the *fitness function* was developed which assigns a value from <0; 1> interval (called *fitness value*) to the particular chromosome. The fitness value is proportional to the quality of the *UUA* transformation (testability factors, chip area overhead and number of I/O pins are evaluated in the fitness function). For better/worse diagnostic properties represented by the chromosome the fitness function is assigned fitness values approximating to 1/0 values. Thus, quality (according to user requirements) of two unique chromosomes (representing unique *UUA* transformations, i. e. solution of the problem) can be compared numerically on the basis of their fitness values.

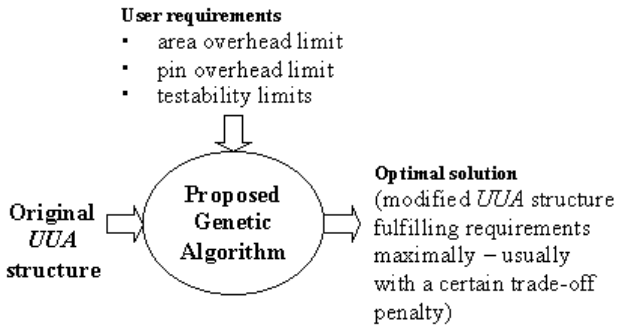


Figure 7. I/O overview of proposed method

Thus, the solution (i. e. scan configuration) we are searching for is represented by a chromosome with a maximal fitness value, i. e. the chromosome with the highest possible fitness value in <0; 1> interval. A genetic algorithm can be used to find such chromosome for given *UUA*.

After the decision on chromosome encoding is done, the crossover comes into play. Crossover selects bits from parent chromosomes and creates a new offspring(s). The simplest way how to do this is to choose randomly some *crossover point* (chromosome bit-position) and everything before this point copy from the first parent to the offspring and then everything after this point copy from the second parent to the offspring.

After the crossover is performed, mutation takes place. This is to prevent solutions to fall into a local optimum of solved problem. Mutation is rarely applied on a new offspring and modifies it randomly. For binary encoded chromosome, mutation can switch a few randomly selected chromosome bits (i. e. from 0 to 1 or from 1 to 0).

The fact that we have a prescription how to describe the *UUA* modification unambiguously by a chromosome, having crossover and mutation functions and having a prescription for assigning a fitness value reflecting the quality of the modification to chromosome, allows us to apply the genetic algorithm to our problem. The genetic algorithm (see bellow for its description) can be understood as a "competition of chromosomes representing various *UUA* modifications" in which the chromosome with the highest fitness value wins.

4.1. Description of the Genetic Algorithm

Let N be the chromosome population size and let P, Q be chromosome sets. The set P will be denoted as the *parent population* and Q set *offspring population*. Then the genetic algorithm consist of the following steps:

1) $P := \emptyset$, $Q := \emptyset$.
2) Generate N chromosomes with random content and store these chromosomes in P set.
3) Assign fitness value to each chromosome $ch \in P$. If any chromosome with a new highest fitness value did not appear in P during the last N iterations or a chromosome $ch \in P$ was generated which satisfies the minimal diagnostic requirements given by a designer or user, then terminate genetic algorithm and return the chromosome $ch \in P$ with a highest fitness value in P as the result of the algorithm (i. e. the solution of the problem), otherwise continue with step 4).
4) From P set select two parent chromosomes p_1, $p_2 \in P$ (the probability of selecting p_1, p_2 chromosomes is proportional to p_1, p_2 fitness values).
5) With the probability of 0,8-0,95 perform crossover.
6) With the probability of 0,005-0,01 perform mutation. Note: the steps 5) and 6) belong to so called reproduction process the output of which are q_1, q_2 offsprings.
7) Store q_1, q_2 into Q set.
8) If $|P| = |Q|$ then perform: $P := Q$, $Q := \emptyset$.
9) Go to step 3).

As a distinctive advantage of utilising genetic algorithm to investigate the search space of chromosomes (compared with other methods used for these purposes) we see the fact that it enables the problem to be defined as a bit string - chromosome and that the algorithm based on genetic algorithm converges rapidly to an optimal (or user-acceptable) solution. The solution is a chromosome identifying *UUA* modification based on an user-acceptable trade-off and implementing selected scan configuration. Let it be noted that in some cases it is not necessary to identify the best chromosome but as an acceptable solution we can see the *UUA* modification which satisfies the user or designer requirements.

5. Experimental Results

We have analysed the state-space size and suggested and implemented a method for optimal solution finding. The state-space analysis goals and principles of our method are presented in previous chapters of this paper.

Because the problem of searching the optimal solution by exploring complete state-space was identified as a combinatorial problem with high time complexity (see t_{all} row in Table 1), there was a need to optimise this problem to reduce the time complexity (see t_{gen} row in Table 1). For optimisation, a method based on genetic algorithm was proposed.

Our method was verified on *DIFFEQ* benchmark circuit and alternative (1) was required (i. e. an order of scan registers in a scan chain). User requirements were: minimal pin and area overheads, maximal testability properties and minimal test application time – it is evident that a trade-off was needed. Experimental results for *DIFFEQ* benchmark circuit are shown in the following two tables (Table 3, Table 4). There are 6 registers in original *DIFFEQ* structure, so $n = 6$. According to Table 1, $nstructs_n$ =8463 and the time t_{all} to gain results fulfilling given criteria by means of *exhaustive search* is 2,3 hour. The purpose of our genetic algorithm is to find the same solution in a shorter time t_{gen}.

In Table 3, a relation among N (population size), number of generations (iterations) of genetic algorithm and t_{gen} (genetic algorithm CPU time valid for PC with PentiumII/333 processor and 64MB RAM) is shown. It can be seen, that all t_{gen} values gained by the proposed genetic algorithm for n=6 are much lower than t_{all} (=2,3h) in Table 1. The higher n the bigger difference between t_{gen} and t_{all} can be seen, but still $t_{gen} << t_{all}$ for $n>>1$. For *DIFFEQ* circuit case (n=6), it can be seen that the number of generations needed to find a solution for different N does not differ substantially, so it can be stated that the number of generations will not decrease considerably with N increasing. Based on experiments with our genetic algorithm, we recommend to use N from 20 to 40 interval, but still such situations can appear in which also other values of N are better than the recommended one. As a solution, two *DIFFEQ* registers (R_1 and R_6) were suggested by genetic algorithm for

modification to scan registers and included into the scan chain configuration denoted by our notation as R_6R_1.

Table 3. Time requirements of proposed genetic algorithm applied on *DIFFEQ* circuit (*n*=6)

	N=20	*N*=40	*N*=60	*N*=80
Number of generations	20	19	18	19
t_{gen} (CPU time for genetic alg.)	7min	9min	11min	14min

In Table 4, columns "Original *DIFFEQ structure*", "Modified *DIFFEQ* structure[1]", "Modified *DIFFEQ* structure[2]" and "*DIFFEQ structure with full scan*" belong to original *DIFFEQ* structure, modified *DIFFEQ* structure based on the solution by means of the proposed genetic algorithm, modified *DIFFEQ* structure based on the solution by means of [15] and *DIFFEQ* structure with built-in full scan (e.g. all 6 *DIFFEQ* registers are modified to scan registers. The resulting scan structure is by our notation denoted $R_6R_5R_3R_1R_4R_2$.

Table 4. Experimental results gained with different methodologies

	Original *DIFFEQ* structure	Modified *DIFFEQ* structure[1]	Modified *DIFFEQ* structure[2]	*DIFFEQ structure* with *full scan*
Total number of PI/PO	11/1	13/2	13/2	13/2
Number PI/PO for test purposes	0/0	2/1	2/1	2/1
Total number of nodes	71	77	83	89
Number/% of controllable nodes	49/69,0%	77/100%	83/100%	89/100%
Number/% of observable nodes	16/22,5%	62/80,5%	66/79,5%	70/78,7%
Average controllability of nodes	0,651	0,978	0,983	0,990
Average observability of nodes	0.197	0.956	0,974	0,983
Average testability of nodes	0.424	0.967	0.979	0.987
Depth (cyclic paths ignored)	12	6	2	2
Structure/max. cycle length	Cyclic/8	Cycle-free/0	Cycle-free/0	Cycle-free/0
Number of combinational gates	599	599	599	599
Number of nonscan FF/gates	48/480	32/320	16/160	0/0
Number of scan FF/gates	0/0	16/224	32/448	48/672
Total number of FF/gates	48/1079	48/1143	48/1207	481271
Gate overhead	0%	5,9%	11,9%	17,8%

[1] modification of *DIFFEQ* structure proposed by genetic algorithm (R_1,R_6 in scan chain)

[2] modification of *DIFFEQ* structure according to [15] (R_4, R_6, R_1, R_5 in scan chain)

It can be seen (see the column No. 1 in Table 4), that unmodified *DIFFEQ* structure needs only 11 primary inputs and 1 primary output, total number of all nodes in this *DIFFEQ* structure is 71 and there is only 599 combinational gates from total amount of 1079 gates. But only 69% of all *DIFFEQ* nodes are controllable and only 22,5% of all *DIFFEQ* nodes are observable. The modified *DIFFEQ* structure is cyclic with maximum cycle length 8 and the depth of this *DIFFEQ* structure is 12.

In the column No. 3, results for *DIFFEQ* structure modified according to [15] are presented. In [15], it is recommended to modify registers R_4, R_6, R_1 and R_5 to scan registers. The resulting scan structure is by our notation denoted $R_4R_6R_1R_5$. It can be seen that two additional primary inputs and one additional primary output are needed for this modification recommended in [15]. The total number of all nodes is 83. The number of combinational gates is still 599, but the total number of gates is 1207 including 448 additional gates for scan registers R_4, R_6, R_1 and R_5. It represents (compared with the column No. 1) gate overhead of 11,9%. But, all nodes in current *DIFFEQ* structure are controllable and about 79,5% of nodes are observable. The modified *DIFFEQ* structure is cycle-free and the depth is only 2.

In the column No. 4, results for *DIFFEQ* with built-in full-scan are presented. It is evident that two additional primary inputs and one additional primary output are needed for full-scan chain. The total number of nodes is 89 which is a higher number than the values in columns No. 1 and 3. The number of combinational gates is still 599, but the total amount of gates is 1271 including 672 additional gates for all 6 scan registers. This fact means (compared with the column No. 1) gate overhead of 17,8%. But, all nodes in current *DIFFEQ* structure are controllable and about 78,7% of nodes are observable. Although there are more registers modified to scan registers than according to column No. 3, the observability is lower than in structures according to the column No. 3 because of higher number of unobservable nodes. The modified *DIFFEQ* structure is cycle-free and its depth is only 2. The resulting scan structure is by our notation denoted $R_6R_5R_3R_1R_4R_2$.

The column No. 2 presents results gained by our method, where only R_1 and R_6 are modified into scan registers. The resulting scan structure is by our notation denoted R_6R_1. The pin overhead is the same as in the case of columns No. 3 and 4, but there are only 77 nodes (6 nodes more than in column No. 1). The number of combinational gates is still 599 and there are 320 non-scan FF gates and only 224 scan FF gates which in total means 1143 gates. The gate overhead in this case is only 5,9% (it is almost 12% lower than in the full-scan case and 6% lower than in [15] case). As in the columns No. 3 and 4, the controllability of all nodes is gained and when compared with columns No. 3 and 4 it can be revealed that there are 80,5% of observable nodes in this structure. The modified *DIFFEQ* structure is cycle-free but there is a trade-off penalty represented by depth=6

It can be seen that except full-scan solution presented in the column No. 4, other methods of selecting registers into scan chains exists. Comparing results from the column No. 2 (proposed by our genetic algorithm) with the results presented by similar methods presented e.g. in [15] (see the column No. 3 of Table 4), [16] or in other one existing approaches it can be said that our method based on genetic algorithm has at least the same

properties of selecting the best solution of this problem. As an advantages of our method we see: 1) much lower time complexity for increasing n (compare t_{all} and t_{gen} values in Table 1) and 2) the ability to identify solutions fulfilling demanded or max available criteria. It can be stated that the methodology presented in this paper can be seen as a completely new approach to the testability improvement process at RT level.

6. Conclusions and Perspectives of Future Research

Not very much attention was devoted so far to the use of evolution algorithms to solve diagnostic problems. This paper is an attempt to show that genetic algorithms can be seen as a tool appropriate for these purposes. The goal of the paper is to show a completely new approach to the partial scan problem. It was presented what types of problems need to be solved when approach is used.

The optimising solution based on genetic algorithm was developed and presented in this paper. In "Experimental results" part of this paper, the CPU time needed to find solutions by means of our method was shown to present the difference between the time complexity of a non-optimised solution and our optimising solution based on genetic algorithm. Also, some of the testability properties of different circuit structures were presented to show a difference between testability properties of a circuit structure based on our genetic approach and the circuit structures based on some other approaches. For the future research, there is a plan for 1) an extension of our optimising method to more test techniques than only the scan technique, 2) experimental results with more benchmark circuits and 3) an improvement of a fitness function. It is also planned to experiment with different approaches to crossover and mutation algorithms.

7. Acknowledgements

This work has been financially supported by the Czech Ministry of Education – grant FRVŠ No. 1754/2002/G1 "Application of Evolution Approaches for Digital Circuit Testability Enhancement", the Grant Agency of Czech Republic grant No. 102/01/1531 "Formal Approaches in Digital Circuit Diagnostics – Testable Design Verification" and the Research Intent of FEI, Brno University of Technology, Czech Republic, grant No. CEZ: J22/98:262200012 "Research of the Information and Control Systems".

8. References

[1] Aktouf, Ch. - Fleury, H. - Robach, Ch.: Inserting Scan at the Behavioural Level, IEEE Design & Test of Computers, vol. 7, No. 3, July - Sept. 2000, pp. 34 - 42

[2] Agrawal, V. D.- Cheng, K. - Johnson, D. D. - Lin, T..: A Complete Solution to the Partial Scan Problem, Proc. of the 1987 International Test Conference, September 1--3, 1987, Washington, pp. 44—51

[3] Higami, Y. - Kajihara, S. - Kinoshita, K.: Partial Scan Design and Test Sequence Generation Based on Reduced Scan Shift Method, Journal of Electronic Testing: Theory and Applications, 7, Kluwer Academic Publishers, 1995, pp. 115--123

[4] Flottes, M. L. - Pires, R. - Rouzeyre, B. - Volpe, L.: A Fast and Effective Technique for Partial Scan Selection at RT Level, Proc. of IEEE ETW 1997, May 28--30, 1997, Cagliary, Italy, pp. 36—42

[5] Abadir, M.S. - Breuer, M.A.: A Knowledge-Based System for Designing Testable VLSI Chips, IEEE Design & Test of C., vol. 2, No. 3, 1985, pp. 56 – 68

[6] Stroele, A. P. - Wunderlich, H. J.: Hardware-Optimal Test Register Insertion, IEEE Trans. On Computer-Aided Design of Integrated Circuits and Systems, vol. 17, No. 6, 1998, pp. 531 - 539

[7] Stroele, A. P. - Wunderlich, H. J.: Test Register Insertion with Minimum Hardware Cost, Proc. ICCAD 95, San Jose, California, USA, pp. 95 -101

[8] Dorsch, R. - Wunderlich, H. J.: Reusing Scan Chains for Test Pattern Decompression, Proc. ETW 2001, Stockholm, May 2001, pp.307 – 315

[9] Chakradhar, S. T. - Balakrishnan, A. - Agrawal, V.: An Exact Algorithm for Selecting Partial Scan Flip-Flops, Journal of Electronic Testing: Theory and Applications, 7, 1995, pp. 83 - 93

[10] Orenstein, T. - Kohavi, Z. - Pomeranz, I.: An Optimal Algorithm for Cycle Breaking in Directed Graphs, Journal of Electronic Testing: Theory and Applications, 7, 1995, pp. 71 - 81

[11] Barbagallo, S. - Bodoni, M. L. - Medina, D. - Corno, F. - Prinetto, P. - Reorda, M. S.: Scan Insertion Criteria for Low Design Impact, IEEE VLSI Test Symposiun, Princeton, April 1996, pp. 26 - 31

[12] Conway, J. H., Guy, R. K.: The Book of Numbers, Springer-Verlag, New York, 1996, 310 pp.

[13] Comtet, L.: Advanced Combinatorics: The Art of Finite and Infinite Expansions, Reidel Publishing Company, Dordrecht, 1974, ISBN 90-277-0380-9, 156 pp.

[14] Sloane, N. J. A., Plouffe, S.: The Encyclopedia of Integer Sequences, Academic Press , Orlando, 1995, 588 pp.

[15] Bukovjan, P.: Allocation en vue de la testabilité dans le cadre de la synthèse de haut niveau, PhD thesis, Institut National Polytechnique de Grenoble, 2000, 138 pp.

[16] Xinli Gu: RT Level Testability Improvement by Testability Analysis and Improvements, PhD thesis, Department of Computer and Information Science, Linköping University, Sweden, 1996, 149 pp.