Evolution of Synthetic RTL Benchmark Circuits with Predefined Testability[1]

TOMAS PECENKA
ON Semiconductor
LUKAS SEKANINA and ZDENEK KOTASEK
Brno University of Technology

This paper presents a new real-world application of evolutionary computing in the area of digital circuits testing. A method is described which enables to evolve large synthetic RTL benchmark circuits with a predefined structure and testability. Using the proposed method a new collection of synthetic benchmark circuits was developed. These benchmark circuits will be useful in a validation process of novel algorithms and tools in the area of digital circuits testing. Evolved benchmark circuits currently represent the most complex benchmark circuits with a known level of testability. Furthermore, these circuits are the largest circuits that have ever been designed by means of evolutionary algorithms. This paper also investigates suitable parameters of the evolutionary algorithm for this problem and explores the limits in the complexity of evolved circuits.

Categories and Subject Descriptors: B.8.1 [**Hardware**]: Performance and Reliability—*Reliability, Testing, and Fault-Tolerance*

General Terms: Design

Additional Key Words and Phrases: benchmark circuit, evolvable hardware, testability analysis

1. INTRODUCTION

Simultaneously with the growing complexity of digital circuits, we can observe the growing complexity of CAD tools that are utilized to produce and verify these circuits. The evaluation and comparison of different algorithms and methodologies utilized in these CAD tools is one of the most difficult tasks CAD users are faced

[1]THIS IS DRAFT, (C) ACM

This research was supported by the Grant Agency of the Czech Republic under contract No. 102/07/0850 “Design and hardware implementation of a patent-invention machine”, No. 102/05/H050 “Integrated Approach to Education of PhD Students in the Area of Parallel and Distributed Systems” and the Research Plan No. MSM 0021630528 “Security-Oriented Research in Information Technology”.
Authors’ addresses: T. Pecenka, ON Semiconductor, Bozeny Nemcove 1720, 756 61 Roznov pod Radhostem, Czech Republic, e-mail: tomas.pecenka@onsemi.com, L. Sekanina and Z. Kotasek, Faculty of Information Technology, Brno University of Technology, Bozetechova 2, 612 66 Brno, Czech Republic, e-mail: {sekanina;kotasek}@fit.vutbr.cz.
Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.
© 2008 ACM -/2008/0700-0001 $5.00

with. Over the years, there have been many attempts to create and use benchmark circuits for evaluation and comparison of these algorithms and methodologies.

For people involved in testing, ISCAS [Brglez and Fujiwara 1985; Brglez et al. 1989] and ITC'99 [Corno et al. 2000b] benchmark circuits are well-known and most commonly used. Unfortunately, the most complex circuit of these sets consists only of 231,320 gates and 6,642 flip-flops. In comparison to the complexity of designs which can be implemented in current ASIC[2] chips, the complexity of existing benchmark circuits is too low. That is especially true for "open-source" benchmark circuits. This problem might be overcome by the utilization of synthetic benchmark circuits.

The synthetic benchmark circuits have been recognized as a viable alternative to the standard benchmark circuits (e.g. [Darnauer and Dai 1996; Iwama et al. 1997; Pistorius et al. 1999; Verplaetse et al. 2002; Hutton et al. 2002; Kundarewich and Rose 2004]). They are usually created by an automated procedure and constrained to have a specific set of desirable characteristics (such as the size, topology, testability and function). In particular, a variable testability and a sufficient complexity of benchmark circuits are crucial features for testing modern CAD test tools. Unfortunately, no approach is known for the design of synthetic benchmark circuits that are both testable and complex.

The objective of this paper is to propose a new method for developing synthetic RTL[3] benchmark circuits with required structure, complexity and testability. The approach is based on the concept of the evolutionary circuit design [Higuchi et al. 1993; Thompson 1998; Miller et al. 2000]. Candidate benchmark circuits are encoded as strings (chromosomes) and an evolutionary algorithm is utilized to perform a search (in the space of digital circuits) for those circuits that maximize a cost function defined by the user.

The idea of evolutionary design of benchmark circuits was initially introduced in [Pecenka et al. 2005]; however, only relatively small circuits were evolved (up to 150,000 gates). Pecenka et al. [2006] described the methodology which allows to create a set of synthetic benchmark circuits with predefined testability. In this paper, it is reported that by careful parameter setting of the evolutionary algorithm, circuits of complexity up to 1.2M can be evolved. Finally, this paper also presents a methodology allowing to validate obtained circuits using a state of the art ATPG[4] tool, FlexTest (Mentor Graphics).

The evolutionary circuit design has allowed engineers to discover novel electronic circuits automatically [Thompson 1998; Miller et al. 2000; Sekanina 2004]. Sometimes the evolved circuits exhibit features that the conventional design approaches have never been able to achieve. On the other hand, only relatively small circuits were evolved automatically so far. Various reasons can be identified why the evolutionary approach is not "scalable" (i.e. it is not able to generate circuits of an arbitrarily increasing complexity). It can be stated that by increasing the number of inputs of a digital circuit by one, the evaluation time doubles (i.e. it grows exponentially), assuming that all possible input combinations are considered in the

[2] Application Specific Integrated Circuit
[3] Register Transfer Level
[4] Automatic Test Pattern Generation

fitness calculation process. A reasonable strategy seems to be to include only a subset of input vectors into a training set; however, for example, for arithmetic circuits some papers show [Miller and Thomson 1998] that evolved circuits do not usually work correctly for the remaining input vectors. If it was possible to completely evaluate a candidate solution in a polynomial time (with respect to the number of circuit inputs and components) the evolutionary design process would be more effective and resulting circuits would be more complex. That is possible in some domains only. For example, image filters consisting of thousands gates were evolved using the so-called functional-level evolution [Sekanina 2004]. In this case, the fitness function does not consider all possible combinations of input pixels; only a representative training image has to be used to obtain a good image filter.

In order to evolve complex benchmark circuits, the choice of a suitable testability evaluation method is a crucial decision. The first option is to use an ATPG tool, generate test sequence for a candidate benchmark circuit and use a Fault Coverage (FC) parameter as the testability measure. An advantage of this approach is that the testability is evaluated very accurately. Unfortunately, the time complexity of ATPG algorithms is exponential. Another option is the use of some of testability analysis (TA) methods. For our purposes, a method with the polynomial time complexity was chosen. The method utilizes the average controllability and observability parameters to estimate the circuit testability [Strnadel 2004]. This paper shows that by combining the polynomial time TA method utilized in the fitness function with an evolutionary search method, benchmark circuits of required testability and complexity can be developed.

Our contribution to the area of automated design of benchmark circuits can be summarized as follows. A methodology was developed which allows synthetic RTL benchmark circuits to be created. The benchmark circuits are expected to be used to verify the effectiveness of algorithms and methodologies which aim at increasing testability parameters of digital circuits and compare different methodologies. As a result of trade-off among testability, area overhead, additional costs, etc., not only highly testable circuits can be developed. Thus, the methodologies to improve testability must be able to process circuits with various levels of testability and provide the user with the recommendations how to improve testability. As an important aspect of our approach, we see the fact that benchmark circuits with different levels of testability can be evolved – based on user requirements in terms of circuit structure, complexity and testability.

To summarize, the existence of the set of benchmark circuits could be an important requisite for those who develop and implement testability improvement methodologies. Based on the use of the benchmark set they will be able to state how their methodology or CAD tool solve the problem of testability for its various initial values.

2. FROM CIRCUIT TESTING TO EVOLUTIONARY CIRCUIT DESIGN

The proposed method combines the concepts of synthetic benchmark circuits, testability analysis and evolutionary design of digital circuits. This section briefly introduces these areas.

2.1 Benchmark circuits

Benchmark circuits are typically used for verification of CAD test tools. Recently, various benchmark sets have been proposed which describe digital circuits at different levels of abstraction. A classification of categories of benchmark sets can be found, for example, in [Harlow 2000]. In the testing community, the following benchmark sets are commonly used:

ISCAS benchmarks. The set originally consisted of 41 gate-level circuits – 10 combinational ISCAS'85 circuits [Brglez and Fujiwara 1985] and 31 sequential ISCAS'89 circuits [Brglez et al. 1989]. In 1993, an Addendum [Gloster 1993] was released that included modifications to some of the original ISCAS circuits and also some new circuits. The ISCAS benchmarks represent a wide variety of problem domains and they are still cited very frequently. Unfortunately, the complexity of the ISCAS circuits is too low; the most complex circuit consists of only 22,179 gates and 1,636 flip-flops [Brglez et al. 1989].

ITC'99 benchmarks. The set was introduced at the International Test Conference in 1999. The set originally consisted of 22 circuits described both at gate and Register Transfer levels [Corno et al. 2000b]. The most complex circuit (b18) contains 68,752 gates and 3,320 flip-flops [Corno et al. 2000b]. The original set was revised in 2002. The most complex circuit (b19) of the second release consists of 231,320 gates and 6,642 flip-flops [URL-ITC99 1999].

ITC'02 benchmarks. The set consists of 12 circuits described at the level of modules [Marinissen et al. 2002]. These circuits are intended to be used for the verification and comparison of methods and tools for modular testing of SoCs (System-on-Chip).

In addition, some other minor benchmark sets and individual circuits are occasionally utilized (e.g. CMU-DSP, Diffeq, Tseng, ...).

During the recent years, the ASIC design flow is rapidly moving from the gate-level towards higher description levels. Most design activities are now performed at the Register Transfer level (RT-level). Unfortunately, the complexity of the existing RT-level benchmark circuits is too small with regard to a potential of current ASIC technology.

2.2 Synthetic benchmark circuits

Recently, the synthetic benchmark circuits have been recognized in some areas as a viable alternative of standard benchmark circuits [Darnauer and Dai 1996; Iwama et al. 1997; Pistorius et al. 1999; Verplaetse et al. 2002; Hutton et al. 2002; Kundarewich and Rose 2004]. Synthetic benchmarks are the circuits created by an automated process and constrained to have a specific set of desirable characteristics. They were initially used by Darnauer and Dai [1996] who utilized them to test the FPGA place and route algorithms. Later, Iwama et al. [1997] have used synthetic benchmarks for testing of logic optimizers. Pistorius et al. [1999] and later Verplaetse et al. [2002] and Stroobandt et al. [2000] have created tools for generating synthetic benchmark circuits for partitioning algorithms testing. Hutton et al. [2002] and Kundarewich and Rose [2004] have developed a tool for the design of sequential benchmark circuits by cloning of existing circuits. The cloned

circuits have been used for testing partitioning, place and route algorithms.

Unfortunately, no approach is known where evolutionary techniques are used to generate synthetic benchmark circuits and also no synthetic circuits with predefined testability are available. In this paper a novel method which utilizes evolutionary algorithm to generate benchmark circuits with predefined testability properties is presented. This kind of circuits would be very useful for verification of CAD tools used in testing.

2.3 Testability analysis methods

In order to automatically generate a circuit with a predefined testability, an algorithm with a reasonable time complexity that is able to evaluate the testability of a digital circuit is needed. Usually, the testability of a circuit is evaluated by means of the controllability and observability parameters. Existing TA approaches differ in the way in which the controllability and observability are defined and measured. In general, the controllability (e.g. of an internal circuit node) is understood as the ability to control the node value from the circuit primary inputs. If it is possible to control the node inputs then such node is called a controllable node. Similarly, a node is referred to as an observable node if a value existing at the node output can be observed at the circuit primary outputs. The goal of the controllability (observability) measures is to evaluate the easiness of controlling (observing) signal values. On the basis of these values, the testability of a circuit is calculated.

In this work, we are interested in TA methods working at RT-level. The first RT-level TA methods (e.g. [Chen et al. 1991; Bhatia and Jha 1994]) have used two-valued evaluation of the circuit nodes (i.e., node is either testable or non-testable). Chen et al. [1991] suppose a node as controllable if there exists a sequence of executable paths in the data flow graph such that after the execution of these paths, the content of the node can take any possible value by adjusting the input values. Bhatia and Jha [1994] introduced utilization of transparency properties of operators (a neutral element is used as one of the operands: zero for additions, one for multiplications, ...) to check the controllability and observability of the circuit nodes. The two-valued evaluation was replaced in [Flottes et al. 1997], where the controllability and observability was evaluated by determination of the probability for justifying and propagating any test data to internal nodes. These probabilities were computed by means of transparency coefficients of operations and the set of elementary data transfers performed between data-path components.

Corno et al. [2000a] introduced three different metrics for taking observability into account during RT-level ATPG. It was demonstrated that using a more exact controllability and observability metric during RT-level test generation improves the attained fault coverage. Fernandes et al. [2004] presented the first non-simulation based approach that efficiently computes controllability of RT-level constructs as a function of the probability distribution at the inputs, but no information about the time complexity of the method was given. Strnadel [2004] has introduced a Virta (Virtual ports Testability Analysis) method which is based on modeling of selected testability properties (the controllability and observability) of circuit components. It allows performing TA of the circuit in the quadratic time complexity with respect to the number of components in the circuit. For the purpose of generating synthetic

benchmark circuits a new TA method called ADFT[5] based on Virta method was developed [Pecenka et al. 2006].

2.4 Evolutionary approaches in the field of testing of digital circuits

In the area of testing, evolutionary algorithms (and genetic algorithms, especially) have been utilized many times in various optimization tasks (for example, [Mazumder and Rudnick 1998]). In case of evolutionary circuit design, Thompson [1998] has initiated a research in evolution of fault tolerant circuits. Later, Garvie and Thompson [2003] have directly evolved simple digital circuits containing a built-in self-test system. Sekanina and Ruzicka [2003] have demonstrated that inherently easily testable image filters can be generated automatically for real-world applications. Lohn et al. [2003] performed functional recovery of a quadrature decoder after a stuck-at-zero fault for a model of an FPGA. Corno et al. [2002] have utilized genetic programming to automatically induce test programs for a microcontroller. In order to perform an autonomous functional self-recovery in FPGAs, Zhang et al. [2005] introduced a population-based consensus scheme to detect and repair faults. Regarding the evolutionary design of benchmark circuits with required testability properties, the initial study was introduced in [Pecenka et al. 2005] and a set of evolved benchmark circuits was presented in [Pecenka et al. 2006].

3. A DESIGN METHOD FOR DEVELOPING BENCHMARK CIRCUITS

A method is proposed which allows to design RT-level benchmark circuits with a predefined testability. The user is supposed to specify the number of primary inputs and outputs of a target circuit, the number of components, the type of components and the required testability. The resulting circuit is in the form of a structurally-described RT-level VHDL[6] code. The evolved circuit typically represents data-path of a complex circuit[7].

3.1 High-level description of evolutionary algorithm

The pseudocode of the proposed evolutionary algorithm is given in Figure 1. An initial N-member population P (typically, $N \approx 20 - 50$) of candidate benchmark circuits (that are encoded as constant-length strings) is generated randomly. A randomly generated circuit consists of components taken from a set of components provided by a user. New populations are formed using a tournament selection (with the base 2) and a mutation operator. The mutation operator modifies M percent ($M \approx 2\%$) of circuit interconnections. The R percent ($R \approx 95\%$) of parents from the previous population are replaced by mutated versions of the parent circuits (2-tournament is used). The elitism is ensured. The evolution is left running for G ($G \approx 100 - 200$) generations.

[5] Automated Design for Testability tool

[6] Very High-Speed Integrated Circuit Hardware Description Language

[7] RTL designs can typically be divided into two parts: a data part and a control part. Testing methods are usually oriented to the data-path testing, because a controller can be modified to be self-testable (see [Hellebrand and Wunderlich 1994]).

```
set time t = 0
create initial population P consisting of N benchmark circuits
evaluate P - perform structural and testability analysis

WHILE (not termination condition) DO
  // create offspring population
  set P' = empty set
  WHILE ( |P'| < R*N) DO
    i = an individual selected using 2-tournament from P
    i' = mutation ( i )
    evaluate i' - perform structural and testability analysis
    insert i' to P'
  END WHILE
  // replacement
  FOR (each individual i1 from P') DO
    i2 = a randomly selected individual from P
    i = 2-tournament between i1 and i2
    replace i2 by i in P
  END FOR
  t = t + 1
END WHILE

convert best individual into VHDL code
```

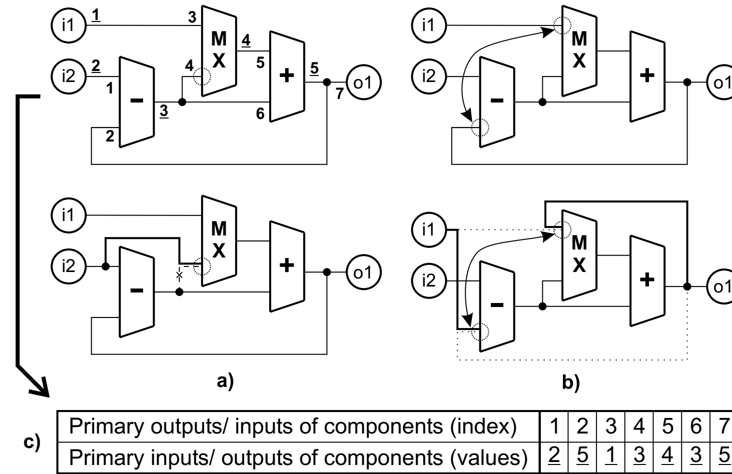
Fig. 1. The principle of the design method

3.2 Circuit representation in the chromosome

A candidate benchmark circuit is considered as a graph consisting of nodes (components) whose inputs and outputs are uniquely numbered. Primary inputs and outputs of the circuit are numbered too. Because 3-state nets are not allowed in resulting circuits, any component input can be connected to only a single component output. As the user specifies the number of components (K) and this number remains constant, every circuit can be represented as a constant-length array in which the index is the component input and the value is the identification of the connected output. The length of chromosome is $\sum_{i=1}^{K} k_i + n_o$, where k_i is the number of inputs of i-th component and n_o is the number of primary outputs. Primary inputs are treated as outputs of a component and primary outputs are treated as inputs of a component connected to the circuit. Figure 2 shows a circuit consisting of three components, its encoding and two options for the mutation operator. Registers are not reflected in the proposed representation; they are inserted into a circuit before the TA is performed.

3.2.1 *Mutation operator.* The mutation operator is applied to modify connections in the circuit under development. It operates in the following way (see Figure 2):

(1) An input of a component (or a primary output) is randomly selected. The mutation operator will be applied on this input.



Primary outputs/ inputs of components (index)	1	2	3	4	5	6	7
Primary inputs/ outputs of components (values)	2	5	1	3	4	3	5

Fig. 2. Two types of mutation (a, b) and circuit representation in chromosome (c)

(2) If the output connected to this input is also connected to another input(s) then the selected input is simply reconnected to a randomly selected output of another component or to one of the primary circuit inputs (see Figure 2a).
(3) If the output connected to this input is not connected to another input(s) then an input of another component (or a primary output) of the circuit under design is randomly selected and the selected inputs are simply reconnected (see Figure 2b).
(4) The mutation operator always maintains the width of the data-path.

The principle of mutation operator is demonstrated in Figure 2. For the first type of mutation (see Figure 2a), an input gate of multiplexer is selected and the connection connected to this gate is simply reconnected to another connection with the same data-path width. For the second type of mutation (see Figure 2b), another input gate of multiplexer was selected. The first type of mutation cannot be used because no other input gate is connected to this connection and the application of this operator will cause the output connected to this connection to become disconnected. Hence, the second type of mutation should be used. It selects a different input gate with the same data-path width and interchanges the connections connected to these inputs.

3.3 Fitness calculation

Obviously, not all randomly generated circuits are valid benchmarks. Before a fitness value is assigned to a candidate circuit, circuit properties are examined. The fitness function, which has to be maximized here, analyzes three features of candidate circuits that we have recognized as important for our method. In particular, circuit structure, interconnections structure and circuit testability have to be inspected.

3.3.1 *Circuit structure analysis.* A candidate circuit is analyzed with the goal to identify isolated sub-circuits. An isolated sub-circuit is a subset of components

which are mutually interconnected, but no output port of any component involved in the isolated sub-circuit is (nor even indirectly) connected to the circuit primary outputs (see Figure 3a). If an isolated sub-circuit is removed then the function of the circuit is not affected. The value of the *structure* parameter is a real number from $\langle 0;1\rangle$ interval:

$$structure = 1 - \frac{useless_comps}{comps_count}, \tag{1}$$

where *useless_comps* denotes the number of components that belong to isolated sub-circuits (i.e. the number of components not affecting circuit behavior) and *comps_count* denotes the number of circuit components.

If an isolated sub-circuit exists in a candidate circuit then the value of *connects* and *testability* parameters (see Equation 4 in next section) is not evaluated and these parameters are set to 0; otherwise, the circuit interconnection and testability analysis follows.

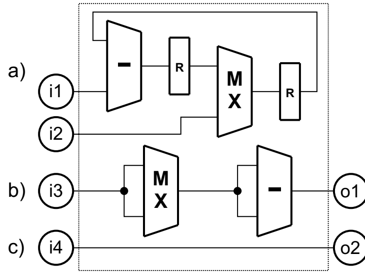


Fig. 3. An example of a circuit with an isolated sub-circuit (a), shorts between inputs (b) and direct connections between primary inputs and outputs(c).

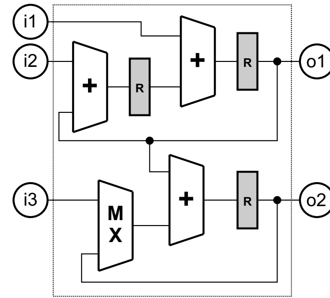


Fig. 4. A circuit with automatically inserted registers

3.3.2 *Circuit interconnection analysis.* The goal of the interconnection analysis is to evaluate the variability of circuit interconnections. Each candidate circuit is analyzed for *shorts* between the inputs that belong to the same component (see Figure 3b) and for direct connections from primary inputs to primary outputs (see Figure 3c). *Comp_inputs* denotes the number of ports of all components and *pri_outputs* denotes the number of circuit primary outputs. The value of *connects* parameter results in a real number from $\langle 0;1\rangle$ interval:

$$connects = 1 - \frac{shorts + direct_connects}{comp_inputs + pri_outputs} \tag{2}$$

3.3.3 *Testability evaluation.* Registers are automatically inserted into the circuit structure before the testability analysis is performed. A register is inserted to the output of each combinational component (except of the multiplexers which are

understood as interconnecting components). If there are more possibilities for register placement, a solution minimizing the number of registers is selected. Figure 4 shows an example of a circuit and registers placement.

In our method, the testability of the circuit is expressed by the controllability and observability parameters. The utilized testability analysis method was formally defined and its time complexity was analyzed in [Strnadel 2004; Pecenka et al. 2006]. The method works as follows:

For the purposes of the testability analysis, two weighted digraphs representing the circuit structure and testability properties are constructed. The first digraph (G_S) represents the test-pattern data-flow and the second digraph (G_I) represents the test-response data-flow. Interface ports of all in-circuit components are treated as vertices of G_S (G_I). An oriented edge exists between two vertices if a test data can be transferred between these vertices, i.e. if a wired connection exists between corresponding in-circuit ports or a component exists which allows to transfer test-vectors (responses) from the input port (first vertex) to the output port (second vertex). These pairs of vertices are put in the relation together with the information about the required flow-condition (e.g. a rising edge of a clock signal, a particular combination of values on other input ports etc.). These relations represent a basis for constructing G_S (G_I) edges.

The TA algorithm is constructed as a graph-searching algorithm over G_S and G_I. During the search process, the accessibility of ports from circuit primary inputs is analyzed in G_S (this step corresponds to the controllability analysis) and the accessibility of ports at circuit primary outputs is analyzed in G_I (this step corresponds to the observability analysis). Each evaluated testability attribute is reflected as a real number from $\langle 0; 1\rangle$ interval, where 0 indicates an absence of this attribute and 1 indicates the occurrence of the attribute in its best form. The evaluation of x-controllability (x means a port of the circuit) can be understood as the evaluation of "easiness of controlling values at x by means of stimuli generated at the circuit primary inputs". Alike, the evaluation of x-observability can be understood as the evaluation of "easiness of observing values at x by means of circuit primary outputs". Because x is understood as testable if it is both controllable and observable, the evaluation of the overall testability is a function of the controllability and observability parameters. An example of the circuit with evaluated controllability and observability parameters (according to [Strnadel 2004]) is given in Figure 5.

It was proven that the algorithm runs in $O(|V(G_S)| \cdot |E(G_S)| + |V(G_I)| \cdot |E(G_I)|)$ time complexity, where $V(G_S)$ is the set of vertices and $E(G_S)$ is the set of edges of test-pattern data-flow digraph G_S [Strnadel 2004]. Similarly, $V(G_I)$ is the set of vertices and $E(G_I)$ is the set of edges of test-response data-flow digraph G_I.

In the fitness function, the results of TA – the average controllability (*avg_cont*) and average observability (*avg_cont*) – are compared with the values of controllability *(req_cont.)* and observability *(req_obs.)* requested by the user. A final value of the *testability* parameter is a real number from $\langle 0; 1\rangle$ interval, which describes the fulfilment of the user requirements, i.e.

$$testability = 1 - 0.5 \cdot (req_cont. - avg_cont)^2 - 0.5 \cdot (req_obs. - avg_obs.)^2 \quad (3)$$

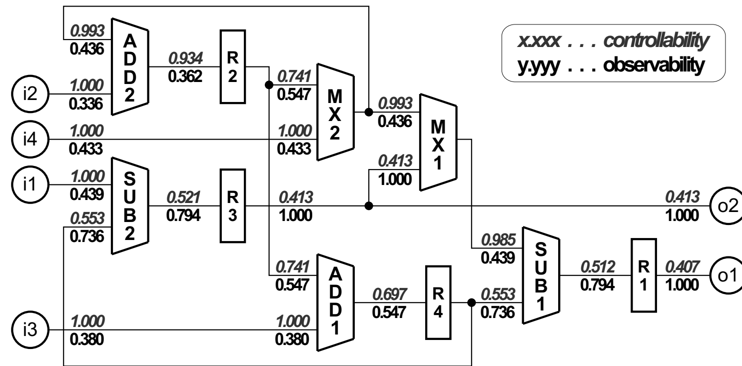


Fig. 5. Example of the circuit with evaluated controllability and observability parameters.

3.3.4 *Overall fitness.* The overall fitness function (given by Equation 4) combines the results of the interconnection analysis and testability analysis (note that the second one is more important here). The resulting fitness value is a real number in $\langle 0; 1\rangle$ interval:

$$fitness = 0.25 \cdot structure + 0.25 \cdot connects + 0.5 \cdot testability. \quad (4)$$

As the result of testability analysis is the most important and results of the interconnection and structure analysis are less important, the weights of coefficients in the fitness function are determined (experimentally) as 1:1:2.

3.4 Implementation of the proposed method

The Cirgen tool was developed which implements the proposed method [Pecenka 2006]. An example of **.xml** code is given which is used by the user to specify the requirements on circuits and testability properties.

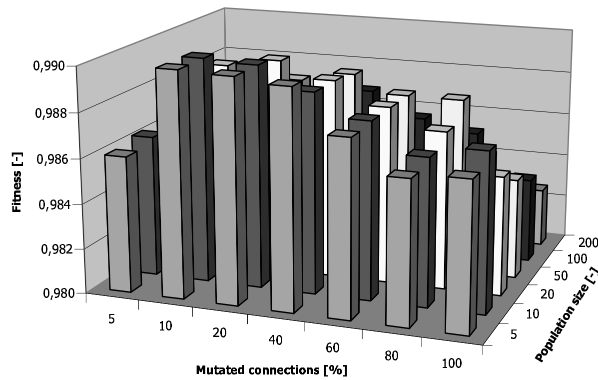
```
<circuit>
  <testability controllability="0.75" observability="0.75"/>
  <evolution population="30" replacement="0.95" mutation="0.02"
  steps="200"/>
  <primary inputs="5" outputs="3"/>
  <comp name="SUB"  width="4" quantity="5"/>
  <comp name="ADD"  width="4" quantity="5"/>
  <comp name="MUX2" width="4" quantity="2"/>
</circuit>
```

From the example it can be seen that 75% controllability and 75% observability is required in average. There are 30 individuals in the mating pool, the mutation probability is 2% and 95% of worst individuals are replaced in each generation. The evolutionary algorithm produces 200 generations. The resulting circuit is expected to have five primary inputs and three primary outputs. It will consist of five 4-bit subtractors, five 4-bit adders and two 4-bit multiplexers, i.e. of 12 components in total. The Cirgen generates the benchmark circuit according to the predefined requirements.

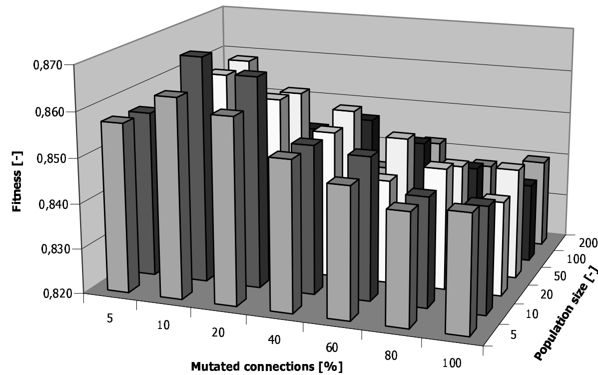
4. EXPERIMENTAL RESULTS

4.1 Searching for suitable parameters of evolutionary algorithm

The objective of the first experiment is to find suitable values of the population size (N) and the mutation rate (M). The experiment is performed with 42 different combinations of the population size ($N = \{5, 10, 20, 50, 100, 200\}$) and the mutation probability ($M = \{5\%, 10\%, 20\%, 40\%, 60\%, 80\%, 100\%\}$). The replacement parameter is fixed to $R = 90\%$ (90% of individuals from the parent population come into tournament with individuals from offspring population). The number of evaluations in each evolutionary run is fixed to 2500 ($G = 500$ for $N = 5$, $G = 125$ for $N = 200, \ldots$) and 90% controllability and observability is required. Firstly, fifteen 50-component circuits were evolved for each combination of parameters. Secondly, ten 500-component circuits were evolved under the same scenario. Figure 6 shows average fitness values of resulting circuits. The best results have been obtained for the population size of 5–20 individuals and the mutation parameter set at 10%–20%.



a) 50-component circuit

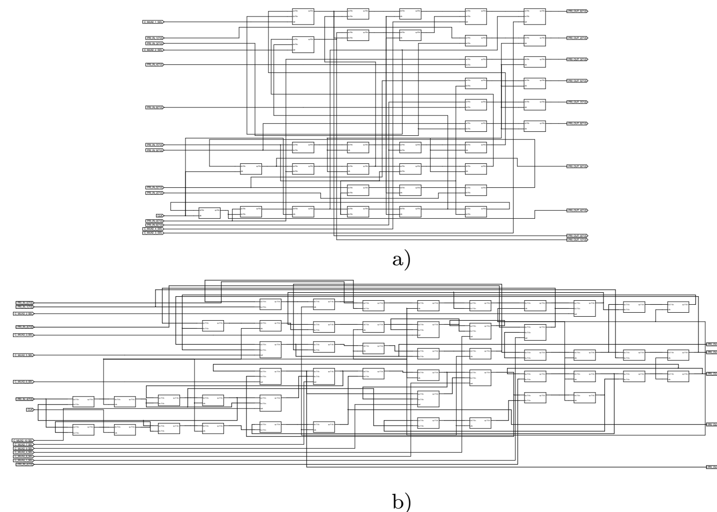


b) 500-component circuit

Fig. 6. Average fitness values of the best individuals for different population sizes and mutation parameters.

4.2 Structure of generated circuits

The objective of the second experiment is to verify the structure, synthesizability and testability of evolved circuits. A circuit consisting of 20 components and high testability and a circuit consisting of 30 components and low testability were evolved with the following parameters: $N = 30$, $G = 200$, $M = 2\%$ and $R = 95\%$ (see Section 3.1 for description). Figures 7(a) and 7(b) show RT-level structure of evolved benchmark circuits. Note that as the purpose of these figures is to show the structure of circuits, the component types and description are irrelevant and thus invisible. The first circuit consists of 20 components and requires 2645 gates after synthesis to TSMC[8] 0.35μ primitives. We required 80% observability and 80% controllability on average; the results obtained by ADFT are 80.6% for observability and 74.5% for controllability. The second circuit consists of 30 components and requires 3605 gates after synthesis to TSMC 0.35μ primitives. We required 20% observability and 33% controllability on average; the obtained results are 20.4% for observability and 36.7% for controllability. Considering a target use of these benchmark circuits, the results are acceptable. Notice that registers were included automatically to the circuits after evolving the circuit structure (see Section 3.3.3), i.e. the figures contain more elements than we have specified.



a)

b)

Fig. 7. Examples of evolved benchmark circuits with high (a) and low (b) testability.

The circuits shown in Figure 7 are structurally different. This difference is caused by different complexity (number of components) and different requirements on testability. The circuit shown in Figure 7(a) (80% controllability and observability required) contains only a few short feedback loops. For the circuit shown in Figure 7(b), 33% controllability and 20% observability were required. It can be seen that the circuit contains many long feedback loops. The structure of these circuits

[8]Taiwan Semiconductor Manufacturing Company Ltd.

matches the well known fact that circuits with many feedback loops are harder to test than those with fewer feedback loops.

4.3 Exploring limits of the method

The objective of this experiment is to test whether the method is able to satisfy user requirements for really complex circuits (containing more than hundred thousands of gates). Experimental setup was left unchanged (see Section 4.2). Experiments were performed for a 50-, 100-, 500- and 1000-component circuit. For all circuits, 33% controllability and observability were required. Table I summarizes the obtained results (averaged from 5 independent runs).

Table I. Complexity and testability of evolved circuits

Requirements		**Results**				
In/Out [-]	RTL comps. [-]	Controllability [%]	Observability [%]	Gates [-]	Flip-flops [-]	Time [hh:mm]
5/5	50	32.76%	32.15%	27,452	2,509	00:10
10/10	100	32.12%	30.42%	55,720	5,176	00:17
40/40	500	33.57%	30.84%	277,674	25,584	02:47
80/80	1,000	34.35%	29.92%	565,193	51,717	11:52

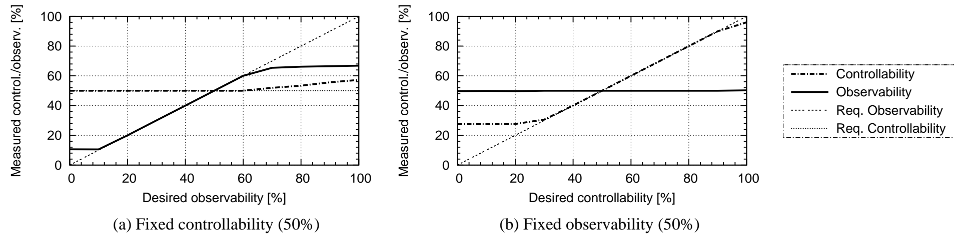
Table I shows that the proposed tool is able to generate circuits up to thousands of RT-level components. It takes several hours to obtain the circuits of this complexity (for a Xeon 2.8GHz CPU, 2GB RAM). The complexity of resulting gate level circuits depends on the complexity of RT-level components provided by the user. The gate level complexities given in Table I are obtained for elementary RT-level components such as 16/32-bit adders, subtractors and multipliers. The method is able to create circuits with the complexity of millions of gates when complex RT-level components are provided.

4.4 Controllability/observability design space exploration

The objective of the next experiment is to explore the design space in case that (a) the controllability is fixed and observability is changed and (b) the observability is fixed and controllability is changed. The experiment was performed for a 50-component circuit with the following requirements:

- 8 primary inputs and outputs.
- 50 components: 8xADD(8bit), 8xSUB(8bit), 8xMUX2(8bit), 8xADD(16bit), 8xSUB(16bit), 8xMUX2(16bit) and 2xMUL(8,16bit).
- The testability requirements on the produced circuits are as follows:
  a) 50% controllability (fixed), the observability 0-100%, incremented with the step of 10%;
  b) 50% observability (fixed), the controllability 0-100%, incremented with the step of 10%.

The average time required for circuit evolution was 9.2 minutes (with Intel Xeon 2.8GHz CPU, 1GB RAM). Results given in Figure 8 are averaged from 20 independent runs. It can be seen that requirements on the controllability and observability



(a) Fixed controllability (50%)

(b) Fixed observability (50%)

Fig. 8. Testability parameters obtained when (a) the controllability is fixed (=50%) or (b) the observability is fixed (50%).

parameters can be satisfied only on a specific range of possible values of these parameters. For example, for the fixed controllability (=50%) and this particular experimental setup, the algorithm is able to satisfy user requirements in the range from 10% to 60% (see Figure 8(a)). For the fixed observability (=50%), the algorithm is able to satisfy user requirements in the range from 30% to 90% (see Figure 8(b)). Consider that these results are strongly influenced by the specification provided by the user (i.e. by the number of circuit primary inputs/outputs, the number of components, testability properties of these components etc.). In general, not all user requirements can be satisfied, simply because desired circuits probably do not exist at all.

4.5 Validation of the proposed method

In the proposed method, the user is supposed to specify the requirements on the circuit structure and on its testability properties. A circuit with "as close structure and parameters as possible" to the specification is sought by the evolutionary algorithm. A set of experiments was performed to analyze whether:

(1) The resulting circuits have a desired structure (i.e. they contain the desired components). A script was created to check circuit structures automatically.
(2) The resulting circuits are synthesizable. Leonardo Spectrum (Mentor Graphics) is used to check synthesizability (note that all additional optimizations provided by this tool are disabled).
(3) The resulting circuits have the desired testability properties. FlexTest ATPG tool is used for the validation of testability.

This analysis was performed on circuits of various complexity (650, 2800, 6500, 13000 and 25000 gates) generated by the Cirgen tool. For each level of the complexity, fifty circuits with various testability were generated with the controllability and observability in the range of 0% to 100%. Their structure as well as synthesizability was evaluated. Then our ADFT tool (see [Pecenka et al. 2006]) was used for the testability analysis and the resulting values were plotted at the x-axis of Figure 9. These circuits were also synthesized to TSMC technology and a corresponding test was generated by FlexTest. The relation between the testability gained by our ADFT tool and the fault coverage parameters measured by FlexTest are shown in Figure 9. The testability of circuits measured by ADFT tool ($testability = (controllability + observability)/2$) is plotted on the x axis and fault coverage of tests generated by FlexTest is plotted on the y axis.

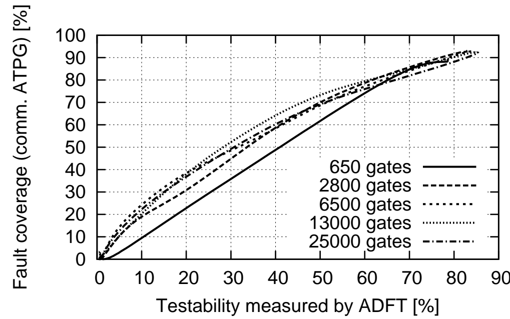


Fig. 9. Relation between the testability measured by ADFT and the fault coverage measured by FlexTest

It can be seen that when a circuit is evaluated as low testable by our ADFT tool, the ATPG tool also indicates a low fault coverage. Similarly, when a circuit is evaluated by ADFT as highly testable, the ATPG tool indicates a high fault coverage. This relation applies to all considered levels of testability. On the basis of these experiments, it can be stated that the TA method implemented in the utilized ADFT tool represents a realistic estimate of testability for the circuit class designed by the proposed method.

5. EXTENDING FITTEST_BENCH06 BENCHMARK SET

The initial implementation of this method was used for developing the FITTest_BENCH06 (Faculty of Information Technology, Brno University of Technology, Testability analysis BENCHmarks) set consisting of circuits e01 – e20 (up to 310,610 gates) [Pecenka et al. 2006]. Determining suitable parameters of the evolutionary algorithm allowed extending this set with four new circuits e21 – e24 and thus reaching the complexity of 1.2M gates. Now the set consists of 35 synthetic sequential circuits. The circuits included into this set can be divided into two groups:

—**Variable complexity/variable testability properties** – 24 circuits with 6 levels of complexity (2,000; 10,000; 28,000; 150,000; 300,000 and 1,200,000 gates). For each level of complexity, circuits at four levels of testability properties exist (the fault coverage measured by FlexTest is approx. 0%; 33%, 66% and 100%). Table II informs about the structure and complexity of all benchmark circuits (table's columns mean: circuit identification, the number of primary gates, the number of flip-flops, the number of logic gates and the fault coverage).

—**Constant complexity/variable testability properties** – 11 circuits with equal complexity (approx. 108,000 gates) but with various testability properties (fault coverage is in the range from 0% to 100%). Table III informs about circuit structure and fault coverage.

An additional information about FITTest_BENCH06 benchmark set (together with the tool which was used to develop it) is available at FITTest_BENCH06 website [Pecenka 2006]. For each benchmark circuit, a detailed information is provided about its structure together with the testability gained by FlexTest. Benchmarks circuits are available both at the RT-level (VHDL code) and gate-level (Verilog, EDIF) in TSMC 0.35μ elements.

Table II. Complexity and testability of FITTest_BENCH06 benchmarks (1st part)

Circuit	PI	PO	# of FFs	# of gates	Fault coverage
e01	86	80	160	1,985	90.45%
e02	86	80	144	1,657	60.69%
e03	86	80	160	2,046	39.43%
e04	86	80	160	2,221	0.00%
e05	186	160	792	10,011	90.11%
e06	186	160	831	9,999	43.90%
e07	186	160	785	9,894	22.87%
e08	186	160	778	9,559	0.00%
e09	211	192	2,020	28,065	91.90%
e10	179	208	1,979	27,853	64.22%
e11	211	200	2,058	28,231	27.46%
e12	203	208	2,106	28,438	0.00%
e13	1,669	1,904	6,304	155,046	89.38%
e14	1,621	1,904	6,368	155,380	64.46%
e15	1,701	1,840	6,368	155,207	31.84%
e16	1,589	1,744	6,368	155,045	12.50%
e17	3,833	4,272	12,672	310,122	81.73%
e18	3,913	4,512	12,608	309,856	56.72%
e19	3,833	4,320	12,576	309,874	40.28%
e20	3,961	4,352	12,736	310,610	23.13%
e21	15,332	17,088	50,688	1,240,488	80.13%
e22	15,652	18,048	59,432	1,239,424	52.33%
e23	15,332	17,280	50,304	1,239,496	38.24%
e24	15,844	17,408	50,944	1,242,440	21.56%

6. DISCUSSION

A common feature of existing evolved circuits is that their complexity is low. Typically, the most complex circuits contain tens of gates (when a perfect functionality is required for all possible input combinations [Miller and Thomson 1998]) or a few thousands of gates (when a satisfactory functionality is required for a chosen subset of input vectors [Sekanina 2004]). The goal of our research was not to develop a methodology enabling to evolve circuits with required function but with predefined testability properties in terms of their structural parameters. With our method-

Table III. Complexity and testability of FITTest_BENCH06 benchmarks (2nd part)

Circuit	# of gates	# of FFs	# of faults	Fault coverage
a00	108 627	4 384	399 806	1,28%
a01	108 748	4 448	399 786	10,11%
a02	108 532	4 416	398 012	23,81%
a03	108 876	4 448	400 166	31,60%
a04	108 551	4 416	399 334	40,38%
a05	108 740	4 448	399 534	50,19%
a06	108 607	4 416	399 494	65,56%
a07	108 811	4 448	399 708	66,37%
a08	108 650	4 448	399 566	74,86%
a09	108 345	4 384	398 888	86,54%
a10	108 652	4 448	399 112	94,29%

ology, it is possible to calculate the estimate of testability for thousands of very complex candidate circuits in reasonable time. Thus, it is possible to evolve large benchmark circuits containing millions of gates. Surprisingly, the complexity limit is not determined by our evolutionary algorithm. There was no reason to evolve more comprehensive benchmarks because the commercial design tools we routinely use for the circuit design were not able to synthesize such large circuits.

Where are the benefits of the proposed method? Firstly, evolved benchmark circuits currently represent the most complex benchmark circuits with a known testability level. These circuits would be useful for testing novel CAD tools intended for the fields such as diagnostics and testing of digital circuits. Secondly, as benchmark circuits are designed by means of the evolutionary algorithm, they can contain constructions which do not usually appear in the circuits designed by classical design techniques that are based on the principles of decomposition and minimization. Thus, the use of evolved benchmark circuits can reveal those problems that remain hidden to conventional benchmark circuits.

A possible objection is that proposed benchmark circuits are not real-world circuits. However, using Cirgen, a user can evolve new benchmark circuits for a specific domain with components that are extracted from real-world designs. Therefore, the fraction of synthetic parts and real-life parts can be partly controlled by the user.

Another issue is related to the register placement algorithm. As far as registers are concerned, they are seen as components through which test is applied and which are easy to be tested if included into scan chains. Test application problems could become stronger if the complexity of combinational blocks between registers increased due to diverse register insertion. Thus, the principles of inserting registers into benchmarks can impact the testability. The delay of combinational circuitry between registers is one of the aspects which can be reflected in the design of benchmark circuits. For this version of Cirgen, we utilized the easiest method – a register is included at the output of each component (except multiplexers).

In fitness function, three criteria are combined using weight coefficients. In the future research we would like to extend the method to be able to solve the problem as a multiobjective optimization problem using the concept of Pareto front. Then,

we will be able to provide a set of different solutions with the same quality instead of a single solution.

7. CONCLUSIONS

A new method for generating synthetic benchmark circuits with required structure and testability was developed. Due to the low time complexity of the utilized TA, the proposed method allows the design of relatively complex circuits (millions of gates) with the required testability and complexity. The testability of resulting circuits was verified by the commercial ATPG tool FlexTest.

The developed method was also utilized to create a set of 35 sequential synthetic benchmark circuits that are available at the RT-level and gate-level. Evolved benchmark circuits currently represent the most complex benchmark circuits with a known level of testability. Furthermore, these circuits are the largest circuits that have ever been designed by means of evolutionary algorithms. The implementation of the proposed method together with the evolved benchmark circuits are available at the Cirgen web site [Pecenka 2006].

REFERENCES

Bhatia, S. and Jha, N. 1994. Genesis: A behavioral synthesis system for hierarchical testability. In *Proceedings of European Design and Test Conference, 1994*. IEEE Computer Society, Paris, France, 272–276.

Brglez, F., Bryan, D., and Kozminski, K. 1989. Combinational profiles of sequential benchmark circuits. In *Proceedings International Symposium on Circuits and Systems (ISCAS)*. IEEE Computer Society, Portland, OR, 1924–1934.

Brglez, F. and Fujiwara, H. 1985. A neutral netlist of 10 combinational benchmark circuits and a target simulator in Fortran. In *Proceedings International Symposium on Circuits and Systems (ISCAS)*. IEEE Computer Society, Kyoto, Japan, 695–698.

Chen, C.-H., Wu, C., and Saab, D. G. 1991. Accessibility analysis on data flow graph: An approach to design for testability. In *Proceedings 1991 IEEE International Conference on Computer Design: VLSI in Computer & Processors*. IEEE Computer Society, 463–466.

Corno, F., Cumani, G., Reorda, M. S., and Squillero, G. 2002. Efficient machine-code test-program induction. In *CEC2002: Congress on Evolutionary Computation*. IEEE, Honolulu, Hawaii, USA, 1486–1491.

Corno, F., Reorda, M. S., and Squillero, G. 2000a. High-level observability for effective high-level ATPG. In *VTS '00: Proceedings of the 18th IEEE VLSI Test Symposium (VTS'00)*. IEEE Computer Society, Washington, DC, USA, 411–416.

Corno, F., Reorda, M. S., and Squillero, G. 2000b. RT-level ITC'99 benchmarks and first ATPG results. *IEEE Design & Test 17,* 3, 44–53.

Darnauer, J. and Dai, W. W.-M. 1996. A method for generating random circuits and its application to routability measurement. In *FPGA '96: Proceedings of the 1996 ACM fourth international symposium on Field-programmable gate arrays*. ACM Press, New York, NY, USA, 66–72.

Fernandes, J. M., Santos, M. B., Oliveira, A. L., and Teixeira, J. C. 2004. A probabilistic method for the computation of testability of RTL constructs. In *DATE '04: Proceedings of the Conference on Design Automation and Test in Europe*. IEEE Computer Society, Washington, DC, USA, 176–181.

Flottes, M. L., Pires, R., and Rouzeyre, B. 1997. Analyzing testability from behavioral to RT level. In *EDTC '97: Proceedings of the 1997 European Conference on Design and Test.* IEEE Computer Society, Washington, DC, USA, 158–164.

Garvie, M. and Thompson, A. 2003. Evolution of combinatonial and sequential on-line self-diagnosing hardware. In *5th NASA / DoD Workshop on Evolvable Hardware (EH 2003)*. IEEE Computer Society, Chicago, IL, USA, 177–183.

Gloster, C. 1993. ISCAS'89 Addendum benchmark set. In *ACM/SIGDA Benchmarks Electronic Newsletter*. ACM.

Harlow, J. 2000. Overview of popular benchmark sets. *IEEE Design & Test of Computers 17,* 3, 15–17.

Hellebrand, S. and Wunderlich, H.-J. 1994. Synthesis of self-testable controllers. In *Proceedings of European Design and Test Conference*. IEEE Computer Society Press, 580–585.

Higuchi, T., Niwa, T., Tanaka, T., Iba, H., de Garis, H., and Furuya, T. 1993. Evolving hardware with genetic learning: A first step towards building a Darwin Machine. In *Proc. of the 2nd International Conference on Simulated Adaptive Behaviour*. MIT Press, 417–424.

Hutton, M. D., Rose, J., and Corneil, D. G. 2002. Automatic generation of synthetic sequential benchmark circuits. *IEEE Transactions on CAD of Integrated Circuits and Systems 21(8),* 8 (8), 928–940.

Iwama, K., Hino, K., Kurokawa, H., and Sawada, S. 1997. Random benchmark circuits with controlled attributes. In *Proc. 1997 European Design and Test Conference*. IEEE Computer Society, Washington, DC, USA, 90–97.

Kundarewich, P. and Rose, J. 2004. Synthetic circuit generation using clustering and iteration. *IEEE Transactions on Computer-Aided Design 23,* 6, 869–887.

Lohn, J. D., Larchev, G. V., and DeMara, R. F. 2003. A genetic representation for evolutionary fault recovery in Virtex FPGAs. In *Evolvable Systems: From Biology to Hardware, 5th International Conference, ICES 2003*. Springer, Trondheim, Norway, 47–56.

Marinissen, E. J., Iyengar, V., and Chakrabarty, K. 2002. A set of benchmarks for modular testing of SOCs. In *Proceedings IEEE International Test Conference (ITC)*. IEEE Computer Society Press, Baltimore, MD, 519–528.

Mazumder, P. and Rudnick, E. 1998. *Genetic algorithms for VLSI Design and Test Automation*. Prentice Hall PTR.

Miller, J., Job, D., and Vassilev, V. 2000. Principles in the evolutionary design of digital circuits – part I. *Genetic Programming and Evolvable Machines 1,* 1, 8–35.

Miller, J. F. and Thomson, P. 1998. Aspects of digital evolution: Geometry and learning. In Evolvable Systems: From Biology to Hardware, Second International Conference, ICES 98. *Lecture Notes in Computer Science 1478*, 25–35.

Pecenka, T. 2006. Brno university of technology: Fittest_bench06 benchmarks & cirgen page. http://www.fit.vutbr.cz/~pecenka/cirgen.

Pecenka, T., Kotasek, Z., and Sekanina, L. 2006. FITTest_BENCH06: A new set of benchmark circuits reflecting testability properties. In *Proc. of 2006 IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop*. IEEE Computer Society, 285–289.

Pecenka, T., Kotasek, Z., Sekanina, L., and Strnadel, J. 2005. Automatic discovery of rtl benchmark circuits with predefined testability properties. In *Proc. of the 2005 NASA/DoD Conference on Evolvable Hardware*. IEEE Computer Society, Los Alamitos, US, 51–58.

Pecenka, T., Strnadel, J., Kotasek, Z., and Sekanina, L. 2006. Testability estimation based on controllability and observability parameters. In *Proceedings of the 9th EUROMICRO Conference on Digital System Design (DSD'06)*. IEEE Computer Society, 504–514.

Pistorius, J., Legai, E., and Minoux, M. 1999. Generation of very large circuits to benchmark the partitioning of FPGA. In *ISPD '99: Proceedings of the 1999 International Symposium on Physical Design*. ACM Press, New York, NY, USA, 67–73.

Sekanina, L. 2004. *Evolvable Components: From Theory to Hardware Implementations*. Natural Computing Series, Springer-Verlag.

Sekanina, L. and Ruzicka, R. 2003. Easily testable image operators: The class of circuits where evolution beats engineers. In *The 2003 NASA/DoD Conf. on Evolvable Hardware*. IEEE Computer Society, Chicago, 135–144.

Strnadel, J. 2004. Testability analysis and improvements of register-transfer level digital circuits. Ph.D. thesis, Brno University of Technology.

Stroobandt, D., Verplaetse, P., and Van Campenhout, J. 2000. Generating synthetic benchmark circuits for evaluating cad tools. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 19,* 9 (9), 1011–1022.

Thompson, A. 1998. *Hardware Evolution: Automatic Design of Electronic Circuits in Reconfigurable Hardware by Artificial Evolution.* Distinguished dissertation series. Springer-Verlag.

URL-ITC99 1999. ITC'99 Benchmarks Web Site. http://www.cad.polito.it/tools/itc99.html.

Verplaetse, P., Stroobandt, D., and Van Campenhout, J. 2002. Synthetic benchmark circuits for timing-driven physical design applications. In *Proceedings of the International Conference on VLSI*, H. Arabnia, Ed. CSREA Press, Las Vegas, Nevada, USA, 31–37.

Zhang, K., DeMara, R. F., and Sharma, C. A. 2005. Consensus-based evaluation for fault isolation and on-line evolutionary regeneration. In *Evolvable Systems: From Biology to Hardware, 6th International Conference, ICES 2005.* Lecture Notes in Computer Science, vol. 3637. Springer, 12–24.

Received May 2007; accepted XXXX