Automatic Discovery of RTL Benchmark Circuits with Predefined Testability Properties

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

Abstract

The paper describes the utilization of evolutionary algorithms for automatic discovery of benchmark circuits. The main objective of the paper is to show that relatively large and complex (benchmark) circuits can be evolved in case that only a given property (e.g. testability) is required and the function of the circuit is not considered. This principle is demonstrated on automatic discovery of benchmark circuits with predefined structural and diagnostic properties. Fitness evaluation for the proposed algorithm is based on testability analysis with linear time complexity. During the evolution, the solutions which are refused to be synthesized by a design system are excluded from the process of developing a new generation of benchmark circuits. The evolved circuits contain thousands of components and satisfy the required testability properties.

1. Introduction

Evolutionary circuit design has allowed engineers to discover novel electronic circuits automatically [14]. Sometimes the evolved circuits exhibit the features that the conventional approaches were not be able to achieve ever. On the other hand, only relatively small circuits were evolved automatically. Many reasons can be identified why the evolutionary approach is not "scalable" (i.e. is not able to generate circuits of arbitrarily increasing complexity). One of the most difficult tasks of the evolutionary circuit design which will be underlined in this paper, is that the large circuits require more time to be evaluated than smaller circuits. Note that in a typical evolutionary algorithm, the number of evaluation (fitness calculations) is given by the product of the population size and the number of generations produced by the algorithm. It is evident that the possibility to explore a larger portion of the search space increases the probability of obtaining a better solution in a given time available for the evolution. As this paper deals with digital circuits, we can mention 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 fitness calculation process. A reasonable strategy seems to be to include only a subset of input vectors into the training set; however, papers such as [9] show that the evolved circuits do not usually work for the remaining input vectors correctly.

If we were able to evaluate completely a candidate solution in a linear time (with respect to the number of circuit inputs/components) the evolutionary design process should be more effective and scalable. Hence we focused our attention on the problems for which the evaluation process requires only a linear time. In particular, some methods exist to predict testability of a design in a linear time.

In order to reduce test generation time and test application complexity, it is important to consider and predict testability of a design. If a test generation and its application related to a particular design is cost-effective, then the design is said to be testable. In this context, we speak about the design property called testability. Usually, testability of a design is estimated and numerically evaluated by a testability analysis (TA) method the goal of which is to detect hard-to-test parts of the design. The proper utilization of TA information can result in testability improvement of the design. Therefore, it is reasonable to evaluate and characterize EDA tools using benchmark circuits that are designed to verify not only the function but also the testability.

A method exists which allows designers to obtain a variety of testability properties (in terms of controllability/observability parameters of circuit internal nodes) in a linear time [13]. The main objective of this paper is to show that relatively large and complex (benchmark) circuits can be evolved in case that only a given testability property is required and the function is not taken into account.

Other objectives were defined as follows: (1) a user can specify the required testability (controllability/observability) properties, (2) a library of internal components with a variety of transparency properties must be available for the process of developing a benchmark circuit, (3) the developed circuits must be synthesizable, i.e. structures which are refused by the design system must be excluded from being involved into resulting evolved structures, (4) the output of the system must be in the form of VHDL code.

In this paper, a new methodology utilizing EA for the process of generating benchmark circuits covering a wide scale of testability properties is proposed. The paper is organized as follows. Section 2 briefly presents basic results of the relevant research in the areas of benchmark circuit design, testability analysis and evolutionary design of digital circuits. In Section 3, the proposed method of the evolutionary design of benchmark circuits is presented. Experiments and results are summarized in Section 4. Conclusions and direction for future research are presented in Section 5.

2. Previous relevant research

Since the proposed method combines the concepts of benchmark circuits, testability analysis and evolutionary design of digital circuits, the state of the art in these areas will be briefly summarized in this section.

2.1. The principles of generating synthetic benchmarks

The level of description of a benchmark circuit depends on its application. For example, the evaluation of high-level synthesis algorithms requires high-level behavioral descriptions of circuits, while routing algorithms can only be tested with low-level physical descriptions. The categories of benchmark circuits are presented in [1]. The benchmark circuits are subdivided into following suites: gate-level test generation (ISCAS85, ISCAS89 suites), high level synthesis (HLSynth89, HLSynth91, HLSynth92 suites), logic synthesis (LGSynth89, LGSynth91, LGSynth93, LGSynth95 suites), physical implementation (Compaction86, PRWorkshop88, Modgen89, LayoutSynth90, PDWorkshop91, LayoutSynth92, PDWorkshop93 suites), circuit simulation (CircuitSim90 suite) and partitioning (Partitioning93 suite).

Many other benchmark suites exist that describe circuits at various levels of abstraction. Information about some popular benchmark suites (ACM/SIGMA, ITC, Politecnico di Torino benchmarks, ...) can be found for example in [5]. However, the existing benchmark suites are not sufficient, since they usually consist of too small circuits and they usually are not very representative for all circuit classes.

Recently, the generation of synthetic benchmark circuits has been recognized as a viable alternative [15, 6, 7]. A major advantage of synthetic benchmarks is that they provide full control over important characteristic parameters, such as size, topological or functional parameters. For each circuit class, different parameters are important in general. The major drawback of synthetic benchmark suites is that it is hard to prove (verify) that a set of circuits is representative for all (or at least a class of) circuits for a given application, since usually not all aspects are modelled in a realistic way. Basically, benchmark validation methods can be divided into two groups: direct and indirect. If the direct validation is used then the selected (i.e. direct) parameters of generated synthetic benchmarks and circuit class representatives are compared. In indirect validation, generated synthetic benchmarks and circuit class representatives are compared by means of a suitable algorithm (e.g. by evaluating the number of connections after placement and routing).

2.2. Testability analysis (TA)

Several TA approaches (some of them are mentioned below) have been proposed and they can be classified according to several aspects. For example, the classification can be based on the abstraction level a TA is supposed to be applied for. Then the following TA approaches can be distinguished: gate-level (e.g. [4]), register-transfer level (e.g. [13]), functional level, behavioral level (e.g. [11]) or multilevel TA approaches. According to TA results application, TA approaches can be divided to general-purpose TA approaches (e.g. [4, 13]) and special-purpose TA approaches (e.g. TA is used for inserting registers into partial scan chains). According to the supposed test type, they can be subdivided into probability-based approaches, deterministic test based approaches, etc.

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

2.3. Evolutionary approaches in the field of diagnostics and testability of digital circuits

Evolutionary approaches to diagnostics and testability of digital circuits have initially been used by Thompson [14] who has tried to evolve fault tolerant circuits. Garvie and Thompson have directly evolved simple digital circuits from the scratch containing a built-in self-test system [3]. Sekanina and Ruzicka have demonstrated that inherently easily testable image filters can be generated automatically for real-world applications [10]. Lohn et al performed functional recovery of a quadrature decoder after a stuck-at-zero fault for a model of the FPGA [8]. Corno et al. have utilized genetic programming to automatically induce test programs for a microcontroller [2]. However, no approach is known for direct evolutionary design of test circuits, i.e. the called benchmarks with predefined testability properties.

3. Design method

We present a novel approach which utilizes an evolutionary algorithm to design a structure of benchmark circuit automatically according to the requirements specified by the user. For indirect validation of generated benchmark circuits the register transfer level testability analysis based on the controllability and observability measurements and on transparency properties of internal components is used. The requirements on the circuit function are not reflected in this procedure.

3.1. Problem definition

Our objective is to produce high quality RT level (Register Transfer level) benchmark circuits automatically. The user is supposed to specify the number of primary inputs and outputs of the circuit, the number and type of components, the requirements on testability (average controllability and observability) and parameters of the evolutionary algorithm.

First it is necessary to define the format of the specification reflecting user requirements. We have decide to use *.xml* description. As an example the following *.xml* code in Figure 1 can be used. We can see that 50% controllability and observability on average is required by the user. There are 30 individuals in mating pool, mutation probability is 2% and 95% of worst individuals are replaced in each generation. The evolutionary algorithm will be run for 200 generations. The resulting circuit is expected to have five primary inputs and outputs and will consist of ten 8-bit subtractors, ten 16-bit adders, ten multipliers, five 8-bit multiplexers and five 16-bit multiplexers, i.e. of 40 components in total.

```
<circuit>
  <testability control="0.5" observ="0.5"/>
  <evolution population="30" replacement="0.95"
             mutation="0.02" steps="200"/>
  <primary inputs="5" outputs="5"/>
  <comp name="SUB" width="8" quantity="10"/>
  <comp name="ADD" width="16" quantity="10"/>
  <comp name="MUL" width="8,16" quantity="10"/>
  <comp name="MUX2" width="8" quantity="5"/>
  <comp name="MUX2" width="16" quantity="5"/>
</circuit>
```

Figure 1. Example of source code

The program generates a benchmark circuit according to the predefined requirements. The resulting circuit consists of components, each of them described behaviorally in VHDL. All the generated circuits are synthesizable. At the moment, the user cannot specify the position of registers. The program inserts the registers automatically in order to meet the requirements on testability and to minimize the number of registers.

3.2. Circuit structure representation

Each circuit is considered as a graph represented by a fixed-size integer array. Evolutionary algorithm operates on these arrays. Registers are not reflected in the representation; they are ”inserted” into a circuit before the testability analysis and synthesis procedures are performed.

A circuit consists of components whose inputs and outputs are uniquely numbered. Primary inputs and primary outputs are numbered too. Because any component input can be connected to only a single component output, we can represent the circuit as an array, in which the index is the component input and the value is the identification of the connected output. Primary inputs are treated as outputs of a component and primary outputs are treated as inputs of a component connected to the testbench circuit. Figure 2 gives a simple example.

3.3. Testability analysis method

An analysis method for testability evaluation of candidate circuit on RT level was developed [12]. The testability of the circuit is expressed by controllability and observability parameters. The controllability parameter expresses the ability to control input ports of circuit components and observability expresses the ability to observe a value of output ports of circuit components. The developed testability analysis method is formally defined. Time complexity of the method was derived and the correctness of the method was formally proved.

For the purposes of testability analysis, two weighted

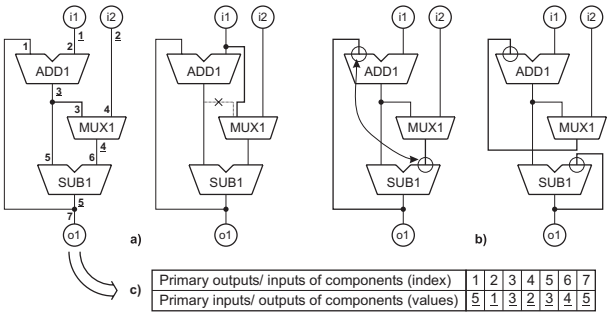


Figure 2. Two types of mutation (a), (b). Circuit representation in chromosome (c)

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. Vertices of G_S (G_I) are interface ports of all in-circuit components. An oriented edge exists between two vertices if a diagnostics data flow is possible between the start-vertex and end-vertex data, i.e. if it is possible to transfer test-vectors (responses) from start-vertex to end-vertex. The pairs of vertices between which test-vectors (responses) data flow is possible are put in the relation together with information about required flow-condition. Those relations form a basis for constructing G_S (G_I) edges.

The proposed testability analysis algorithm is constructed as a graph-searching algorithm over G_S and G_I. During the search process, accessibility of ports from circuit primary inputs is analyzed in G_S (this step corresponds to controllability analysis) and accessibility of ports at circuit primary outputs is analyzed in G_I (this step corresponds to observability analysis). Each measured diagnostic attribute is evaluated (using a proper formula) by a real number from $\langle 0; 1\rangle$ interval, where 0 indicates an absence of this attribute and 1 indicates occurrence of the attribute in its best form. The evaluation of *x*-controllability (*x* means a port of the circuit) can be understood as evaluation of "easiness of controlling values at *x* by means of stimuli generated at circuit primary inputs". Alike, evaluation of *x*-observability can be understood as 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, evaluation of testability is a function of controllability and observability.

Some of important properties of proposed testability analysis algorithm were proven in [13] - especially the algorithm correctness and the time complexity. 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. 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.

3.4. Evolutionary algorithm

An evolutionary algorithm operating with the representation introduced in Section 3.2 was used. The initial population consisting of *P* individuals is generated randomly. New populations are formed using tournament selection and mutation operator. The *N* weakest candidate circuits are replaced by mutated parents, elitism is ensured. The evolution is left to run for a given number of generations. The fittest individual is considered as an acceptable result and is transformed to VHDL code.

In our methodology of developing benchmark circuits, the mutation operator is applied to modify connections in the circuit under development. It means that if a connection exists in the circuit then as a result of mutation the connection is reconnected.

The principles can be summarized in the following way (see Figure 2):

1. The input of a component on which the mutation operator will be applied is selected.
2. If the output connected to this input is also connected to another component(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 component(s) then another input of a component or primary output of the circuit under design is randomly selected and selected gates are simply reconnected (see Figure 2b).
4. The mutation operator always respects the size of data path (i.e. it maintains the width of the data path).

3.5. Fitness calculation

The fitness function, which has to be maximized here, combines three objectives: (1) circuit structure, (2) component interconnections and (3) circuit testability.

At first, the circuit structure is evaluated. The circuit is analyzed with the goal to identify isolated subcircuits (the isolated subcircuit is a subcircuit which is not connected to other parts of the benchmark circuit under design) and for structures which are possibly to be removed during synthesis (see Equation 1).

$$structure = 0.25 \cdot \left(1 - \frac{useless\_comps}{comps\_count}\right) \qquad (1)$$

where *useless_comps* denotes the number of components in isolated subcircuits and the number of components to be removed during the process of synthesis and *comps_count* denotes the number of all circuit components. Value of *structure* parameter is a real number from $\langle 0; 0.25 \rangle$ interval.

If isolated subcircuits and structures which could be removed are found, the fitness evaluation process is stopped and the value of *connects* and *testability* parameters is set to 0. If no separated subcircuits and structures which could be removed are found in the circuit, the circuit interconnection and testability analysis follows.

The goal of interconnection analysis is to evaluate the variability of interconnections of the circuit (see Equation 2). The circuit is analyzed for components whose inputs are connected to the same point (*short_cuts*) and for direct connections from primary inputs to primary outputs (*direct_connects*). *Comp_inputs* denotes a sum of all components inputs and *pri_outputs* denotes the number of circuit primary outputs. The value of $connects$ parameter is evaluated by a real number from $\langle 0; 1 \rangle$ interval.

$$connects = 1 - \frac{short\_cuts + direct\_connects}{comp\_inputs + pri\_outputs} \quad (2)$$

The testability analysis follows after interconnection analysis. The testability parameters are calculated using the algorithm presented in Section 3.3 (see [13] for details). An algorithm is able to evaluate average controllability *(avg_cont.)* and observability *(avg_obs.)* of a candidate circuit in linear time complexity. The values are compared with the values of controllability *(req_cont.)* and observability *(req_obs.)* required by the user. The value of $testability$ parameter is a real number from $\langle 0; 1 \rangle$ interval.

$$\begin{aligned} testability &= 1 - 0.5 \cdot (req\_cont. - avg\_cont)^2 - \\ &\quad 0.5 \cdot (req\_obs. - avg\_obs.)^2 \end{aligned} \quad (3)$$

The fitness function given by Equation 4 combines the results of components interconnections and testability analysis. The result value is a real number from $\langle 0; 1 \rangle$ interval.

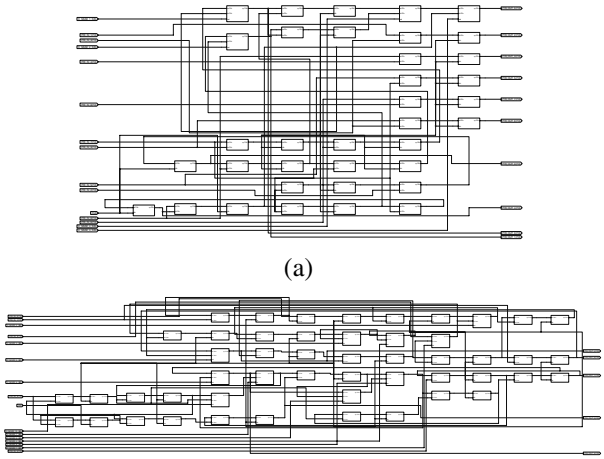
$$\begin{aligned} fitness &= structure + 0.25 \cdot connects + \\ &\quad 0.5 \cdot testability \end{aligned} \quad (4)$$

4. Experiments and results

We have performed hundreds runs of the evolutionary design process in order to find suitable parameters of the evolutionary algorithm. Typical experiments are as follows.

4.1. Overall testing

Figures 3(a) and 3(b) show the examples of benchmark circuits developed with our methodology. The first circuit consists of 20 components and requires 83 Virtex slices after synthesis to Xilinx Virtex II FPGA. We required 80% observability and 80% controllability on average; the obtained results are 80.6% for observability and 74.5% for controllability. The second circuit consists of 30 components and requires 139 Virtex slices after synthesis to Xilinx Virtex II FPGA. We required 20% observability and 33% controllability on average; the obtained results are 20.4% for observability and 36.7% for controllability. Notice that registers were included to the circuits after the design of the structure of the circuits, i.e. the figures contain more elements.



(a)

(b)

Figure 3. Examples of evolved benchmark circuits: 20-component circuit (a) and 30-component circuit (b)

The circuits in Figure 3 are structurally different. These differences are caused by specified testability requirements. For circuit in Figure 3(a) 80% controllability and observability was required. It can be seen that the circuit contains only a few short feedback loops. For circuit in Figure 3(b) 33% controllability and 20% observability were required. It can be seen that the circuit contains many long feedback loops. The structure of circuits matches the theory that circuits with many feedback loops are hard to test (some testabitily improvement methods are focused on long feedbacks loops splitting).

4.2. Evolvability of the produced circuits and meeting the requirements

The objective of this task was to observe how the maximal fitness value (gained from several independent runs) increases during the evolution and how observability and controllability of the evolved benchmark circuits differ from the values required by the user. In other words, we investigated whether the produced solution is improved over time. This experiment was performed for a small circuit (20 components) and a large circuit (500 components).

20-component circuit. Evolutionary algorithm was used with the following parameters: population size was 40 individuals, 200 generations were produced, mutation probability was 2% and replacement probability was 95%. Results were obtained as average value from 20 independend runs. The structural requirements on the produced circuits were: 80% controllability, 80% observability, 5 inputs, 5 outputs, 20 components: 8xADD(8bit), 8xSUB(8bit) and 4xMUX2(8bit).

500-component circuit. Evolutionary algorithm was used with the following parameters: population size was 40 individuals, 200 generations were produced, mutation probability was 2% and replacement probability was 95%. Results were obtained as average value from 10 independend runs. The structural requirements on the produced circuits were: 80% controllability, 80% observability, 40 inputs, 40 outputs, 500 components: 80xADD(8bit), 80xADD(16bit), 80xSUB(8bit), 80xSUB(16bit), 80xMUX2(8bit), 80xMUX2(16bit) and 20xMUL(8,16bit).

Figures 4 and 5 show that the average fitness value (gained from 20 (a) and 10 (b) independent runs) is continually improved over time, i.e. the quality of the solutions does not stagnate on average.

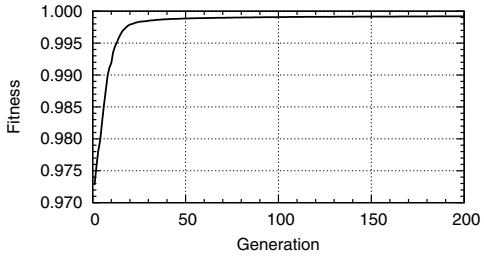


Figure 4. Evolution of a 20-component circuit.

Figure 6 shows the differences of required and obtained controllability and observability values for 20 evolved 20-component circuits. The obtained controllability ranges between 79.35% and 79.82% on average. The obtained observability ranges between 76.22% and 77.46% on average.

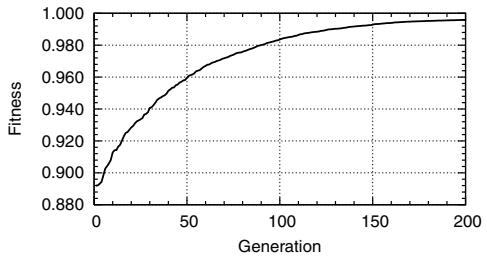


Figure 5. Evolution of a 500-component circuit.

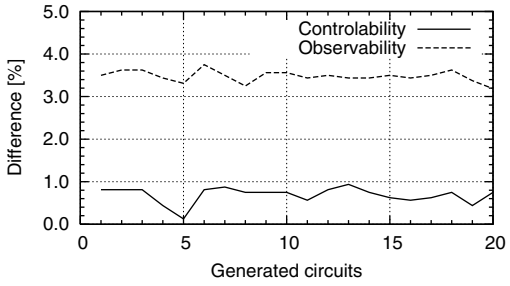


Figure 6. The differences of required and obtained controllability and observability for 20 evolved 20-component circuits.

Figure 7 shows the differences of required and obtained controllability and observability for 10 evolved 500-component circuits. The obtained controllability ranges between 77.02% and 78.82% on average. The obtained observability ranges between 72.99% and 75.18% on average.

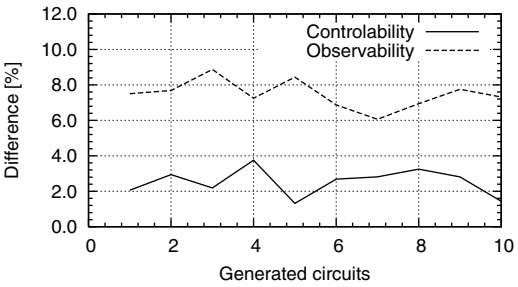


Figure 7. The differences of required and obtained controllability and observability for 10 evolved 500-component circuits.

The average time required for the evolutionary design of circuit was 6.7 minutes for 20-component circuit and 9.4 hours for 500-component circuit on Intel P4-2.6GHz with 512MB RAM.

The developed program is able to generate circuits up to size of thousand of RT components. The time needed to generate circuits of such complexities is about days (Intel P4-2.6GHz with 512MB RAM). The complexity of resulting gate level circuits depends on complexity and datawidth of used components. The complexity of evolved gate level circuits can be very high if library components which represent complex circuits on RT level are used instead of elementary RT level components. In the developed method the user is allowed to add new library components. Therefore, various circuits with required complexity/testability properties can be generated automatically.

4.3. Controllability/observability design space exploration

The objective of this task was to utilize the proposed evolutionary method to explore a part of the design space in case that controllability (observability) is changed and observability (controllability) remains a constant.

This experiment was performed for a 50-components circuit. Evolutionary algorithm was used with the following parameters: population size was 30 individuals, 200 generations were produced, mutation probability was 2% and replacement probability was 95%. Results were obtained as average value from 20 independend runs. The structural requirements on the produced circuits were: 8 inputs, 8 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 were: (a) 50% controllability (fixed), observability is incremented with the step of 10%; (b) 50% observability (fixed), controllability is incremented with the step of 10%.

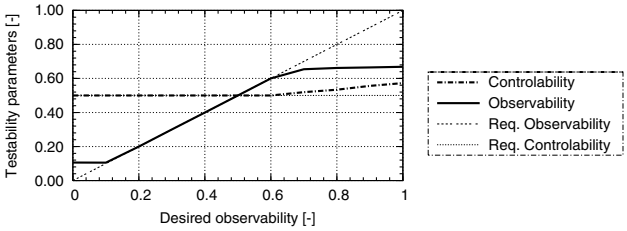


Figure 8. Testability parameters obtained when observability increases from 0.0 to 1.0 and controllability is fixed (=0.5).

The average time required for circuit generation was 34.6 minutes (Intel P4-2.6GHz, 512MB RAM). It can be seen in Figure 8 and Figure 9 that requirements on controllability and observability parameters of circuit could be satisfied

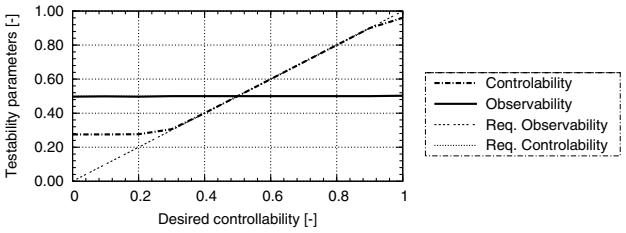


Figure 9. Testability parameters obtained when controllability increases from 0.0 to 1.0 and observability is fixed (=0.5).

only on a specific range of possible values of controllability and observability parameters. This restriction is caused by structural properties of the circuit (the number of circuits primary inputs/outputs, the number and diagnostic properties of used components). For example, the structure of circuit does not allow creating circuits with the 0% controllability (observability) parameters because the 0% controllability (observability) circuit corresponds to a circuit with none controllable input (observable output).

5. Conclusions

In our research we have verified that useful benchmark circuits can effectively be evolved. The evolved circuits are relatively complex; so far no other methodology in the evolvable hardware field was used to evolve so large circuits. The reason is that we have not dealt with functionality of the circuits. As we have analyzed the testability only, we were able to evaluate a candidate circuit in the linear time with respect to the number of components. When functionality is evolved, the fitness calculation time grows exponentially with the number of components (and inputs).

Thus, the evolved circuits have certain diagnostic properties. The aspect of function covered by the circuit was not seen important at the moment and was not taken into account.

In the future, the research will be focused on verification of relation between controllability and observability parameters and testability of the circuit measured by the number of circuit test vectors generated by the commercial ATPG (Automated Test Pattern Generation) tool and to the improvement of the testability analysis method. Testability analysis based on controllability and observability parameters evaluation is always performed for the complete circuit now. However, each circuit contains parts which were not modified when compared to the last evaluation. Therefore, it is not needed to analyze these parts again. The utilization of the testability analysis method which is able to analyze only the modified parts of the circuit (incremental testability

analysis) could reduce the time needed for fitness evaluation and thus more complex circuits could be evolved.

5.1. Acknowledgements

This work has been financially supported by the Grant Agency of the Czech Republic under contract No. 102/04/0737 "Modern Methods of Digital Systems Synthesis" and by the FRVŠ foundation under contract No. 3041/2005/G1 "Evolutionary Design of Benchmark Circuits".

References

[1] Computer-Aided Design Benchmarking Laboratory. http://www.cbl.ncsu.edu/benchmarks.

[2] F. Corno, G. Cumani, M. S. Reorda, and G. Squillero. Efficient machine-code test-program induction. In *Congress on Evolutionary Computation*, pages 1486–1491, 2002.

[3] M. Garvie and A. Thompson. Evolution of self-diagnosing hardware. In A. Tyrell, P. Haddow, and J. Torresen, editors, *Proc. of 5th Int. Conf. on Evolvable Systems (ICES 2003): From Biology to Hardware*, volume 2606 of *LNCS*, pages 238–248, Trondheim, Norway, 2003. Springer-Verlag.

[4] L. H. Goldstein and E. L. Thigpen. Scoap: Sandia controllability/observability analysis program. In *DAC '80: Proceedings of the 17th conference on Design automation*, pages 190–196. ACM Press, 1980.

[5] J. Harlow. Overview of popular benchmark sets. *IEEE Design & Test of Computers*, 17(3):15–18, 2000.

[6] M. Hutton, J. Rose, and D. Corneil. Automatic generation of synthetic sequential benchmark circuits. *IEEE Transactions on CAD*, 21(8):928–940, 2002.

[7] P. D. Kundarewich and J. Rose. Synthetic circuit generation using clustering and iteration. In *Proceedings of the 2003 ACM/SIGDA eleventh international symposium on Field program-mable gate arrays*, pages 245–247, 2003.

[8] J. Lohn, G. Larchev, and R. DeMara. A genetic representation for evolutionary fault recovery in virtex fpgas. In *Proceedings of the 5th International Conference on Evolvable Systems: From Biology to Hardware*, pages 47–56, Trondheim, Norway, 2003.

[9] J. F. Miller and P. Thomson. Aspects of digital evolution: Geometry and learning. *Lecture Notes in Computer Science*, 1478:25–35, 1998.

[10] L. Sekanina and R. Rika. Easily testable image operators: The class of circuits where evolution beats engineers. In *The 2003 NASA/DoD Conf. on Evolvable Hardware*, pages 135–144, Chicago, 2003.

[11] S. Seshadri and M. Hsiao. Behavioral-level dft via formal operator testability measures. *Journal of Electronic Testing*, 18(6):596–611, 2002.

[12] J. Strnadel. Normalized testability measures based on rtl digital circuit graph model analysis. In *Proceedings of the 5th International Scientific Conference Electronic Computers*, pages 200–205, Košice, 2002.

[13] J. Strnadel. *Testability Analysis and Improvements of Register-Transfer Level Digital Circuits*. PhD thesis, Faculty of Information Technology, Brno University of Technology, Brno, 2004.

[14] A. Thompson. *Hardware Evolution: Automatic design of electronic circuits in reconfigurable hardware by artificial evolution*. Distinguished dissertation series. Springer-Verlag, 1998.

[15] P. Verplaetse, D. Stroobandt, and J. Van Campenhout. Synthetic benchmark circuits for timing-driven physical design applications. In H. Arabnia, editor, *Proceedings of the International Conference on VLSI*, pages 31–37, Las Vegas, Nevada, USA, 6 2002. CSREA Press.