Metrol. Meas. Syst., Vol. XVIII (2011), No. 1, pp. 00-00

PAN POLSKA AKADEMIA NAUK

METROLOGY AND MEASUREMENT SYSTEMS

Index 330930, ISSN 0860-8229

www.metrology.pg.gda.pl

M&MS

TWO HEURISTIC ALGORITHMS FOR TEST POINT SELECTION IN ANALOG CIRCUIT DIAGNOSES

Andrzej Pułka

Silesian University of Technology, Faculty of Automatic Control, Electronics and Computer Science, Institute of Electronics, Akademicka 16, 44-100 Gliwice, Poland (✉ andrzej.pulka@polsl.pl, +48 32 237 1644)

Abstract

The paper presents a heuristic approach to the problem of analog circuit diagnosis. Different optimization techniques in the field of test point selection are discussed. Two new algorithms: SALTO and COSMO have been introduced. Both searching procedures have been implemented in a form of the expert system in PROLOG language. The proposed methodologies have been exemplified on benchmark circuits. The obtained results have been compared to the others achieved by different approaches in the field and the benefits of the proposed methodology have been emphasized. The inference engine of the heuristic algorithms has been presented and the expert system knowledge-base construction discussed.

Keywords: analog circuit diagnosis, fault diagnosis, testing, search methods, heuristic optimization.

1. Introduction

The complexity of electronic systems has been growing very rapidly for the last few decades now and modern devices are often placed on a single chip with hundreds of pins. Because of that huge density the area of a design is usually limited and additional pins for testing have influence on the overall costs. So, the problem of *design for testability* (DfT) [1] is one of the most important factors of system quality. The paper addresses the technique of proper (optimal) selection of the tests points, which is very important during the design process. The section 2 explains the motivation of the work and briefly recalls the background of testing methodologies based on a fault dictionaries; section 3 introduces two searching algorithms: SALTO [2] and COSMO (a modified and extended version of [3]) and an inference environment of the expert system in PROLOG; section 4 presents benchmark examples [4] and makes a generalization the methodology to complex devices; section 5 describes the implementation and the inference engine, and section 6 concludes the paper and emphasizes the benefits of the proposed approach.

2. Motivations against the background of related works

The problem of appropriate selection and minimization of the number of measuring nodes belongs to one of the crucial tasks of chip manufacturing and is strongly demanded. This operation can reduce the production costs, make the circuit easily diagnosable and eliminate design bugs at early production stages. The approach presented in [5] proposes a heuristic inclusion procedure of test point selection based on the concept of ambiguity sets [6]. However, this technique is not efficient and very time consuming, which excludes its application to big circuits. The work described in [1] introduces faster optimization procedures based on integer code sorting and heuristic approach. Methods presented in [7, 8]

Article history: received on Oct. 23, 2010; received in revised form on Feb. 2, 2011; accepted on Feb. 28, 2011; available online on Mar. 21, 2011.

apply the information theory and entropy index to include the test node selection procedure. The entropy-based approach allows to obtain optimal (minimal) sets of test points, however it requires relatively complex mathematical evaluations (logarithms), which are time- and resource-consuming in case of hardware implementation. The techniques based on genetic programming [9] are slow; they require many iterations to obtain a final solution, which is not acceptable in a large circuit. Authors of [10] present a very interesting approach based on directed acyclic graphs search. They claim that the test point selection problem is not a permutation problem but rather a combinational problem with the final complexity:

$O(Number\_of\_faults \times Number\_of\_graph\_nodes \times \log(Number\_of\_faults))$.

This approach, which in fact is similar to decision diagrams, seems to be interesting, but almost every decision requires checking of many conditions. And finally, one of the very recent papers in the field [11] introduces another heuristic technique based on discrete particle swarm optimization (DPSO) and multidimensional fitness function (MDFDPSO).

The main goal of the paper is to show that it is possible to radically reduce the searching space[12, 13], and even utterly replace complicated engines [9, 10, 14, 15] employed in the process of finding the solution with a quite simple and natural mechanism. The presented approach will be called 'commonsense' rather than 'heuristic' because it tries to mimic the natural behavior of an experienced tester, whose choices are based on observations of the environment.

2.1. Background

The theory of analog circuits testability [1, 6, 16] distinguishes two main categories of the test scenarios: *simulation-before-test* (SBT) [17] and *simulation-after-test* (SAT). On the other hand, we can split testing methodologies [1] into: fault-driven testing that investigates faults in a *circuit-under-test* (CUT) and *specification-driven-testing* analyzing parameters fluctuations of CUTs.

If we assume that a given design process meets DfT requirements, the access to the CUT is limited by a process of test points selection. In [18] the author extends the term *test points* and suggests to consider not only circuit nodes, but also voltage levels, frequency tests, parameters describing a signal shape, sampling of the response signal etc., which are necessary to separate the circuit states expressed by states of this test points. In the subsequent paragraphs test points are nodes and their values correspond to nodal voltages (potentials). The fundamental diagnostic methods of analog systems are based on fault dictionaries [5, 6, 7, 14, 19]. The fault dictionary is generated for a given CUT before the test by a set of simulations of potential faults (including faulty-free case). The fault dictionary consists of all measurements – states of test points of a given circuit. It allows separating all possible faults. To properly construct a fault dictionary [6] we have to plan a minimal set of measurements (tests) with stimuli sets that allows identifying subsequent faults. The fault dictionary can be compared to a database or even knowledge-base $\boldsymbol{S}$ of the CUT where every element of the circuit state (fault number; where S_0 corresponds to the faulty-free state):

$$\boldsymbol{S} = \{S_0, S_1,.., S_N\}, \tag{1}$$

is represented by a vector of node states (in the presented examples potentials):

$$N_t = \{n_{1t}, n_{2t},.., n_{Kt}\}. \tag{2}$$

Because of the fact, that over 90% of all possible faults occurring in practical circuits are of the catastrophic nature [20] (i.e. open or short circuits) and the other approaches [9, 10, 15] consider such faults, we will also focus only on catastrophic faults.

The continuous nature of analog signals and the problem of fluctuation of circuits parameters and they tolerances are reflected in *measurement ambiguity*, which means that results of some measurements are so close to one another that they are not possible to be separated, so they create *Ambiguity Sets* (AS) for the tested circuit states. This concept was introduced first in [6] and then it was widely accepted by other researchers [5, 12, 14, 21]. The ambiguity group is defined as any two faulty conditions that fall into the same ambiguity set if the gap between the voltage values produced by them is less than 0.7 V [5, 9].

Very common are *integer-coded dictionaries* [5, 14] and diagnostic techniques based on them can be compared to signal quantization, where a given range of the input signal values is represented by integer code. The fault dictionary obtained in such a way is a two-dimensional vector, where first dimension refers to the fault number, while the other represents the circuit states coded with integers. Integer-coded fault dictionaries proved to be very effective for the optimum test points selection [7, 9, 19, 21, 22].

We need to define the circuit *diagnosability* – the ability to uniquely separate every fault.

Definition 1: A given CUT represented by the integer-coded fault dictionary $\boldsymbol{D} = \{\boldsymbol{S}, \boldsymbol{N}\}$ is *diagnosable* iff for every fault $S_i \in \boldsymbol{S}$ there exists at least one unique vector of node states $N_t \subset \boldsymbol{N}$ ($N_t = \{n_{1t}, n_{2t},.., n_{Kt}\}$) that *only* identifies (separates) fault S_i – $\boldsymbol{SEP}(S_i, N_t)$ i.e. there is no other fault S_j which has a signature corresponding to N_t; where: set of faults: $\boldsymbol{S} = \{S_0, S_1,.,. S_N\}$; set of nodes states: $\boldsymbol{N} = \{N_{t1}, N_{t2},.,. N_{tL}\}$ and set of dictionary elements: $\boldsymbol{D} = \{(S_i, N_t,)| S_i \in \boldsymbol{S} \wedge N_t \subset \boldsymbol{N}\}$. So, formally we have:

$$\underset{\boldsymbol{S_i} \in \boldsymbol{S}(0 \le \boldsymbol{i} \le \boldsymbol{N})}{\forall} \quad \underset{\substack{\boldsymbol{N_t} \subset \boldsymbol{N} \\ (\boldsymbol{S_i}, \boldsymbol{N_t}) \in \boldsymbol{D}}}{\exists} \quad \boldsymbol{SEP}(\boldsymbol{S_i}, \boldsymbol{N_t}) \Leftrightarrow \neg\left(\exists \boldsymbol{S_j} \in \boldsymbol{S}(0 \le \boldsymbol{j} \le \boldsymbol{N}, \boldsymbol{j} \ne \boldsymbol{i}) \Rightarrow (\boldsymbol{S_j}, \boldsymbol{N_t}) \in \boldsymbol{D}\right). \quad (3)$$

We can formulate optimization task as follows: *find a minimal number of measurements (points) that allow uniquely separating entire set of faults*. There are many approaches to this minimization [9, 10, 11, 19, 22] and all of them point out that the exhaustive search is time consuming and belongs to NP-hard problems.

3. The optimization algorithms

The test points selection has been classified [14] as a task of *data mining and information reduction*, i.e. searching for unrecognized relationships between elements hidden inside the database. Prasad and Babu [21] distinguish two techniques: inclusive and exclusive. The algorithms addressed in the next section belong to the inclusive methods, but the exclusive technique can be implemented at the end of the second algorithm (complex) when searching for redundancies. The first, simple algorithm tries to model the common-sense behavior of a typical testing engineer who is to find a fault. The complicated mathematical evaluations in the entropy search are replaced with sorting nodes and looking for 'neighbor' (i.e. placed adjacently in the dictionaries) faults signatures. Only simple comparisons and selections are to be done and the procedure allows finding all absolutely necessary measuring nodes and, in case that they cover entire set of assumed faults, selecting minimal set of test points. The second algorithm comes from various heuristic search techniques [12, 21] and it allows finding at least almost optimal solutions (i.e. those that contain optimal sets).

The procedure of test points selection consists of two main algorithms SALTO and COSMO. The both algorithms (procedures) work under the expert system control, first the SALTO algorithm (actually step 1 of the COSMO) is invoked, and then if necessary the COSMO procedure starts. Fig. 1 gives the general view of the methodology.

Algorithm SALTO (Simple Algorithm Looking for Test pOints)

Step 1: *Initialize an empty list of selected test points:* $\boldsymbol{Sel}_0 = \emptyset$ *and the list of undiagnosed faults that contains all circuits states:* $\boldsymbol{U} = \boldsymbol{S}$.

Step 2: *Check if there exists any node that single separate (diagnose) faults, i.e. the state of a given node is unique for a fault. If yes, remember this 'singleton sets' for each node:*

$$\left(S_i \in \boldsymbol{Single}(n_m) \wedge (S_i, N_i) \in D\right) \Leftrightarrow \underset{N_k \subset N}{\forall} \left(N_k \neq N_i \Rightarrow n_{mk} \neq n_{mi}\right).$$

Step 3: *Put nodes in order according to their importance (ambiguity set size per node, availability etc.):*

$$\{n_1, n_2, \cdots n_K\} \rightarrow \{n_{j1}, n_{j2}, \cdots n_{jK}\} \quad \wedge \quad \overline{\overline{n_{j1}}} \geq \overline{\overline{n_{j2}}} \geq \cdots \geq \overline{\overline{n_{jK}}}.$$

Step 4: *Sort elements of the dictionary according to their signatures evaluated by nodes order and nodes states:*

$$\{N_{t1}, N_{t2}, \cdots N_{tL}\} \rightarrow \{N_{v1}, N_{v2}, \cdots N_{vL}\} \quad \wedge \quad N_{v1} \leq N_{v2} \leq \cdots \leq N_{vL}.$$

Step 5: *Find a distance between signatures of every two neighbor faults, i.e. faults arranged adjacently within the sorted dictionary (with adjacent signatures):*

$$\underset{\substack{(S_m, N_{tm}) \in D \wedge (S_n, N_{tn}) \in D \\ N_{tm} \neq N_{tn}}}{\forall} d_{mn} = N_{tm} - N_{tn} = \sum_i^N \delta_i \quad where: \quad \delta_i = \begin{cases} 0 & for \quad n_{im} = n_{in} \\ 1 & for \quad n_{im} \neq n_{in} \end{cases}.$$

Step 6: *Take the first (the most significant) node and check if any two neighbor faults differ only in this node. If yes, add this node to the list of selected test points and perform step 7. If not, take the next node from the list and try again step 6* (*in case the list is empty go to step 9*):

For i = j_1 **to** j_N **if** U ≠ ∅ **then**

do {**if** *for any* $S_m, S_n \in \boldsymbol{U}$ ($d_{mn} = 1 \wedge n_{im} \neq n_{in}$) **then** $\boldsymbol{Sel_i = Sel_{i-1} \cap \{n_i\}}$ $\boldsymbol{U} - \{S_m, S_n\}$ and **goto** Step 7}

else goto Step 9.

Step 7: *Find if there are any faults in the 'singleton set' for the selected node and remove them from the list of undiagnosed faults:* $\forall \left(S_i \in \boldsymbol{Single}(n_{jm}) \wedge (S_i, N_t) \in D\right) \quad \boldsymbol{U}\text{-}\{S_i\}$.

Step 8: *Find if there are any faults in the list of undiagnosed faults, but diagnosable by nodes of the current list of selected nodes. If yes, remove them from the list of undiagnosed faults. Take the next node from the list and if it is not an empty list, try again step 6:*

$$\underset{\boldsymbol{S_k} \in \boldsymbol{D}}{\forall} \; \boldsymbol{if} \underset{\substack{\boldsymbol{N_{tk}} = \{\boldsymbol{n_{atk}}, .., \boldsymbol{n_{btk}}\} \\ \{\boldsymbol{n_a}, ..., \boldsymbol{n_b}\} \in \boldsymbol{Sel_k}}}{\forall} \boldsymbol{SEP}(\boldsymbol{N_{tk}}, \boldsymbol{S_k}) \quad \boldsymbol{then}: \quad \boldsymbol{U} = \boldsymbol{U} - \{\boldsymbol{S_k}\}$$

Step 9: *Check the list of undiagnosed faults. If it is empty, terminate the program with the result of list of selected nodes. If the list is not empty, try another method* [9, 22] *or the COSMO algorithm with the initial sets* (*reduced search space*) *produced by the SALTO.*

Algorithm COSMO (COmplex Searching MethOd) (modified [2])

Step 1: *Run the SALTO algorithm and find the initial set of nodes.*

Step 1a: *If the list of undiagnosed faults is empty (*$\boldsymbol{U} = \emptyset$*), terminate the search process.*

Step 2: *Group the undiagnosed faults (circuit states) according to the signatures given by the selected nodes, i.e. faults belonging to the same set have the same signature of previously selected nodes:*

$$\boldsymbol{Findall} \quad \boldsymbol{G_k} \quad \boldsymbol{that} \underset{\substack{\boldsymbol{S_i}, \boldsymbol{S_j} \in \boldsymbol{U} \\ \boldsymbol{S_i} \neq \boldsymbol{S_j}}}{\forall} \boldsymbol{S_i} \in \boldsymbol{G_k} \wedge \boldsymbol{S_j} \in \boldsymbol{G_k} \quad \underset{\boldsymbol{n_a} \in \boldsymbol{Sel_k}}{\forall} \boldsymbol{n_{ai}} = \boldsymbol{n_{aj}}.$$

Step 3: *Check the distance between every two faults belonging to the same set. If you find that there is only one node allowing their separation add it immediately to the set of selected nodes and update the knowledge-base (perform the same procedures as those in steps 6, 7 and 8 of the SALTO algorithm).*

$$Findall \quad n_c \quad that \underset{G_k}{\forall} \underset{\substack{S_i, S_j \in G_k \\ S_i \neq S_j}}{\forall} \underset{n_c \in Sel_k}{\exists} n_{ci} \neq n_{cj} \underset{\substack{N_{tk} = \{n_{atk}, .., n_{btk}\} \\ \{n_a, ..., n_b\} \in Sel_k}}{\forall} N_{tk}^{'} = \{n_{atk}, .., n_{btk}, n_{ctk}\}$$

$$\Downarrow$$

$$SEP(N_{tk}^{'}, S_i) \wedge SEP(N_{tk}^{'}, S_j) \quad so \quad Sel_{k+1} = Sel_k \cup \{n_c\} \wedge U = U - \{S_i, S_j\}.$$

Step 3a: *If the list of undiagnosed faults is empty, terminate the search. Otherwise repeat steps 2 and 3 until you observe that the number of remaining faults has not decreased. In such case go to step 4.*
Step 4: *If there are only two undiagnosed faults (trivial case), find the list consisting of all remaining nodes which have different values.*
Step 5: *Try*[*)] *to add other nodes taking into account other factors* (*for example, the most or the least frequently appearing on the difference list etc.*). *After every updating of selected nodes, go to step 2. Examine all sets that fully cover the entire fault dictionary and separate sets with the minimum number of elements*.
Step 6: *Try*[*)] *to minimize the selected sets of minimal number of elements using the exclusive approach; try to take the common part of all sets and find redundant elements with method of "tries"*.
Step 7: *Terminate the program with the list*(s) *of selected nodes.*

Remarks: We can cancel steps 2–3 of the SALTO algorithm, assuming that this information comes from the simulation phase and is known before the start of the optimization. In relation to the COSMO methodology, the type of the employed searching strategy [12] depends on additional information concerning circuit structure, signals graph, terminal nodes, direct neighborhood of elements (nodes) etc. This additional data may be given at the beginning of the program run or could be supplied interactively by a user if necessary (if the inference engine asks about it). Such a solution complicates the automation of the expert system, but it enables the user to keep control under the testing procedure.

[*)] Steps 5 and 6 – contain an imprecise description "Try", which refers to the employment of FDL rules based on additional information, additional factors reflecting circuit structure etc. (in AI it is often called ignorance).

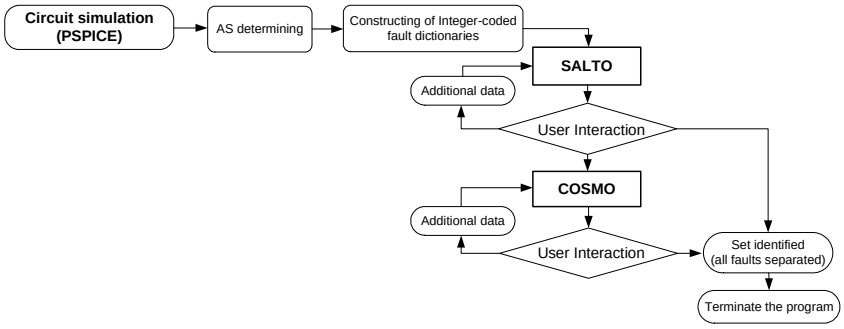

Fig. 1 The general idea of the diagnostic expert system.

4. The experimental tests of the algorithms

At first, the presented approach has been tested on the example of the analog filter found in the literature [9, 10, 22], to compare the results with previous works. Also, the same set of faults and integer codes of testing points (for V_{in} = 4V and f_{in} = 1kHz) has been used to investigate the same dictionary. The SALTO algorithm has produced optimal results in 10 ms! [2]. Then other benchmark examples [4, 8] have been tested. These circuits as well as the first example belong to the set of benchmarks available via Web sites [23]. Very interesting information delivers the comparison of results of three kinds of filters: state-variable, leap-frog and elliptical (Fig. 2–4).

Tables 1-3 present the dictionaries containing the faults grouped by the AC signatures of nodes. The filters have been excited with the voltage signal V_{in} = 3V and f_{in} = 1kHz and every catastrophic fault (short and open circuits) for each discrete element within the circuit has been simulated. Appropriate comments (element symbol with the keyword "*short*" or "*open*") are attached to the states descriptions within the tables. Voltages of a given circuit nodes have been observed and gathered into the rows. Then, the dictionaries have been coded with integer numbers according to the fundamental principles recalled in section 3. The number of AS is minimal and reflects the worst case considerations. The tables show that this AC experiment does not allow separating every fault, but allows distinguishing different circuit states. If a

given state corresponds to more than one fault an additional measurement is required to locate the actual fault.

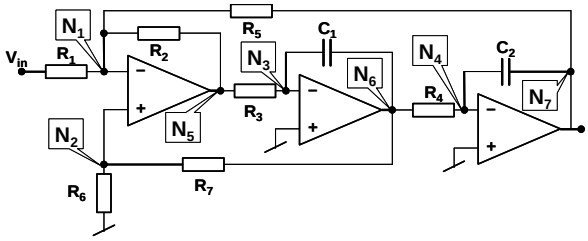

Fig. 2 State-Variable Filter (Benchmark Circuit #1).

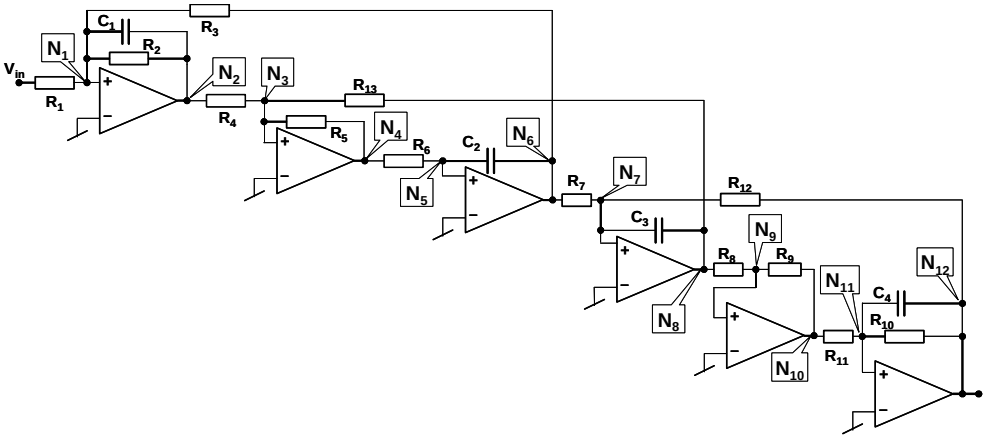

Fig. 3 Leapfrog Filter (Benchmark Circuit #2).

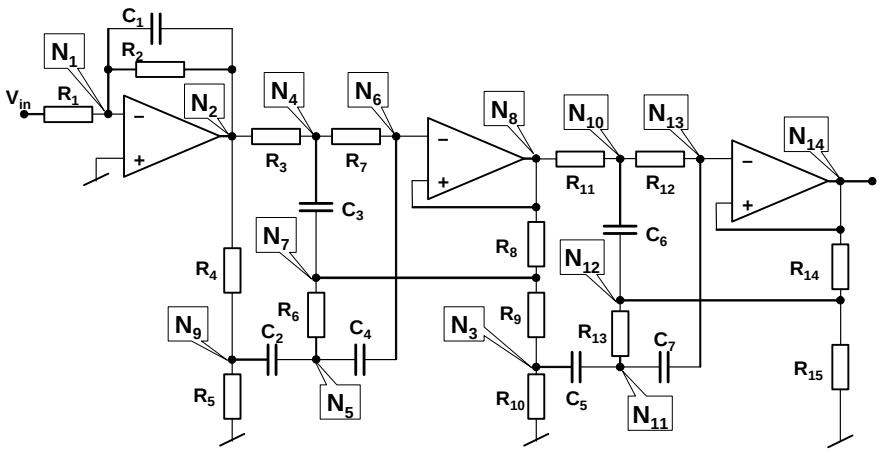

Fig. 4 Elliptical Filter (Benchmark Circuit #3).

Table 1. State-Variable Filter Fault Dictionary.

	Node Nr:	1	2	3	4	5	6	7
1.	{FF, S3} (faulty-free, R2 open)	1	1	0	0	4	3	3
2.	{S1, S4} (R1 open; R2 short)	0	0	0	0	0	0	0
3.	S2 (R1 short)	2	2	0	1	6	5	6
4.	S5 (R3 open)	1	0	0	0	0	0	0
5.	S6 (R3 short)	1	1	0	0	0	2	2
6.	S7 (R4 open)	0	0	1	0	3	0	0
7.	{S8, S17} (R4 short, C2 open)	0	0	0	0	0	0	4
8.	S9 (R5 open)	1	1	0	0	3	2	2
9.	S10 (R5 short)	0	0	0	0	3	0	0
10.	{S11, S14} (R6 open; R7 short)	1	1	0	0	1	1	1
11.	{S12, S13} (R6 short, R7 open)	0	0	0	0	5	4	5
12.	S15 (C1 open)	1	1	0	0	0	1	2
13.	S16 (C1 short)	0	0	0	0	2	0	0
14.	S18 (C2 short)	1	1	0	0	2	1	0

Table 2. Leap-Frog Filter Fault Dictionary.

Node Nr:	1	2	3	4	5	6	7	8	9	10	11	12
1. faulty-free	0	1	0	2	0	1	0	1	0	1	0	1
2. {S1, S4, S6, S28} (R1 open, R2 short, R3 short, C2 short)	0	0	0	0	0	0	0	0	0	0	0	0
3. S2 (R1 short)	1	4	1	5	2	5	1	4	1	4	1	4
4. S3 (R2 open)	0	3	0	4	0	4	0	3	0	3	0	3
5. S5 (R3 open)	0	2	0	3	0	3	0	2	0	2	0	2
6. {S7, S10, S26} (R4 open, R5 short, R13 short)	0	2	0	0	0	0	0	0	0	0	0	0
7. S8 (R4 short)	0	0	0	3	0	3	0	2	0	2	0	2
8. {S9, S27} (R5 open, C1 open)	0	1	0	1	0	1	0	1	0	1	0	1
9. S11 (R6 open)	2	4	0	5	1	5	1	4	1	4	1	4
10. {S12, S29} (R6 short, C2 open)	0	1	0	0	0	1	0	1	0	1	0	1
11. {S13, S24, S32} (R7 open, R12 short, C3 short)	0	2	0	2	0	2	0	0	0	0	0	0
12. S14 (R7 short)	0	2	0	0	0	0	0	2	0	2	0	2
13. {S15, S18} (R8 open, R9 short)	0	1	0	3	0	3	0	2	0	0	0	0
14. {S16, S17} (R8 short, R9 open)	0	2	0	2	0	2	0	0	0	2	0	2
15. S19 (R10 open)	0	2	0	1	0	1	0	1	0	1	0	3
16. {S20, S21, S34} (R10 short, R11 open, C4 short)	0	1	0	3	0	3	0	2	0	2	0	0
17. S22 (R11 short)	0	2	0	2	0	2	0	0	0	0	0	2
18. S23 (R12 open)	0	1	0	3	0	3	0	2	0	2	0	2
19. S25 (R13 open)	0	2	0	2	0	2	0	2	0	2	0	2
20. S30 (C2 short)	0	2	0	2	0	0	0	0	0	0	0	0
21. S31 (C3 open)	0	2	0	1	0	1	0	1	0	1	0	1
22. S33 (C4 open)	0	1	0	2	0	2	0	1	0	1	0	1

Table 3. Elliptical Filter Fault Dictionary.

	Node Nr:	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1.	{FF, S24, S26} (faulty-free, R2-o.)	0	1	1	3	2	2	2	2	1	2	2	2	2	2
2.	{S1, S4, S32} (R1-o., R2-s., C1-s.)	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3.	S2 (R1 short)	2	2	4	8	6	7	6	7	3	4	4	4	4	5
4.	S3 (R2 open)	0	1	0	3	2	2	2	2	0	0	0	0	0	0
5.	S5 (R3 open)	0	1	0	0	0	0	0	0	0	0	0	0	0	0
6.	S6 (R3 short)	0	1	2	6	3	5	5	5	0	3	3	3	3	4
7.	{S7, S10} (R4 open, R5 short)	0	1	1	3	2	3	3	3	0	2	2	2	2	2
8.	{S8, S9} (R4 short, R5 open)	0	1	2	4	5	5	5	5	4	3	3	3	3	4
9.	S11 (R6 open)	0	1	1	3	1	2	2	2	1	1	1	1	1	2
10.	S12 (R6 short)	0	1	2	4	4	4	4	5	1	3	2	3	3	3
11.	S13 (R7 open)	0	1	0	3	1	1	1	1	0	0	0	0	0	0
12.	S14 (R6 short)	0	1	1	4	3	4	4	4	3	3	2	3	3	3
13.	S15 (R8 open)	0	1	1	3	3	1	0	1	1	1	1	1	1	2
14.	S16 (R8 short)	0	1	2	3	2	3	3	3	1	2	1	2	2	2
15.	S17 (R9 open)	0	1	0	3	2	3	3	3	1	3	2	3	3	3
16.	S18 (R9 short)	0	1	2	3	1	2	2	2	1	1	1	1	1	1
17.	S19 (R10 open)	0	1	3	3	2	3	3	3	1	2	2	2	2	2
18.	{S20, S22} (R10 short, R11 short)	0	1	0	3	2	2	2	2	1	2	2	2	2	3
19.	{S21, S23, S27} (R11-o.,R12-o.,R14-s.)	0	1	1	3	2	2	2	2	1	1	1	1	1	1
20.	S25 (R13 open)	0	1	1	3	2	2	2	2	1	2	1	1	1	1
21.	{S28, S29} (R14 short, R15 open)	0	1	1	3	2	2	2	2	1	1	1	2	2	2
22.	S30 (R15 short)	0	1	1	3	2	2	2	2	1	1	1	0	1	1
23.	S31 (C1 open)	1	2	3	7	4	6	5	6	2	3	1	0	1	1
24.	S33 (C2 open)	0	1	2	5	3	4	3	4	1	2	1	0	1	1
25.	{S34, S38} (C2 short, C4 short)	0	1	1	3	1	1	1	1	1	1	1	0	1	1
26.	S35 (C3 open)	0	1	2	4	2	3	3	3	1	2	1	0	1	1
27.	S36 (C3 short)	0	1	1	1	1	1	1	1	1	1	1	0	1	1
28.	S37 (C4 open)	0	1	2	6	4	5	5	5	1	2	1	0	1	1
29.	S40 (C5 short)	0	1	1	4	2	2	2	2	1	0	1	0	1	1
30.	{S39, S41, S42, S44} (C5 o., C6 o., C6 s., C7 s.)	0	1	1	4	2	2	2	2	1	1	1	0	1	1
31.	S43	0	1	1	4	2	2	2	2	1	1	1	0	2	2

o. – open circuit, s. – short circuit.

```
SALTO algorithm started
Ordered Nodes: [7,5,6,2,1,4,3]
New order of faults after the sorting procedure run: [2,7,13,9,11,4,12,10,14,5,8,1,3,6]
Numbers of selected measuring nodes: [7].
List of remaining faults: [2,4,5,6,8,9,12,13,14]     Total number of unrecognized faults equals 9
Numbers of selected measuring nodes: [5,7]
List of remaining faults: [2,4,5,6,9,12,13,14]       Total number of unrecognized faults equals 8
End of Node search. Selected nodes: [5,7]
COSMO algorithm started
Node nr: 1 has to be added.     Node nr: 6 has to be added.     Node nr: 3 has to be added
Numbers of selected measuring nodes: [1,3,5,6,7]
List of remaining faults: []                 Total number of unrecognized faults equals 0
```

Fig. 5 Results of the analysis of the SV Filter Fault Dictionary (Table 1).

```
SALTO algorithm started
Ordered Nodes: [6,4,12,10,8,2,5,1,11,9,7,3]
Numbers of selected measuring nodes: [6]
List of remaining faults: [1,2,3,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22]
Total number of unrecognized faults equals 21
Numbers of selected measuring nodes: [4,6]
List of remaining faults: [2,3,5,6,7,8,9,11,12,13,14,15,16,17,18,19,21,22]
Total number of unrecognized faults equals 18
Numbers of selected measuring nodes: [4,6,12]
List of remaining faults: [2,3,5,6,7,8,9,13,14,16,17,18,19,21]
Total number of unrecognized faults equals 14
Numbers of selected measuring nodes: [4,6,10,12]
List of remaining faults: [2,3,5,6,7,8,9,14,18,19,21]          Total number of unrecognized faults equals 11
End of Node search            Selected nodes: [4,6,10,12]

COSMO algorithm started
Node nr: 2 has to be added     Node nr: 8 has to be added
Numbers of selected measuring nodes: [2,4,6,8,10,12]
List of remaining faults: [3,9]            Total number of unrecognized faults equals 2
Because there are only 2 unrecognized faults: 3 and 9
to separate them you should add one of the following nodes: [1,3,5].
```

Fig. 6 Results of the analysis of the LF Filter Fault Dictionary (Table 2).

```
SALTO algorithm started
Ordered Nodes: [8,6,4,7,5,14,13,12,11,10,9,3,2,1]
Numbers of selected measuring nodes: [4]
List of remaining faults: [1,2,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,25,26,28,29,30,31]
Total number of unrecognized faults equals 27
Numbers of selected measuring nodes: [2,4]
List of remaining faults: [1,4,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,25,26,28,29,30,31]
Total number of unrecognized faults equals 25
End of Node search            Selected nodes: [2,4]

COSMO algorithm started
Node nr: 10 has to be added
Numbers of selected measuring nodes: [2,4,10]
List of remaining faults: [1,4,7,8,9,10,11,12,13,14,16,17,18,21,22,25,29,31]
Total number of unrecognized faults equals 18
Node Nr: 3 has been added
Numbers of selected measuring nodes: [2,3,4,10]
List of remaining faults: [1,4,7,8,9,10,11,13,21,22,25,29,31]
Total number of unrecognized faults equals 13
Node Nr: 14 has been added
Numbers of selected measuring nodes: [2,3,4,10,14]
List of remaining faults: [1,4,7,9,11,13,21,22,25]
Total number of unrecognized faults equals 9
Node Nr: 8 has been added
Numbers of selected measuring nodes: [2,3,4,8,10,14]
List of remaining faults: [9,21]
Total number of unrecognized faults equals 2
Because there are only 2 unrecognized faults: 9 and 21
to separate them you should add one of the following nodes: [5,12,13]
```

Fig. 7 Results of the analysis of the Elliptical Filter Fault Dictionary (Table 2).

Fig. 5, 6 and 7 present results obtained for the state-variable, leap-frog and elliptical filters, respectively. The SALTO algorithm is not sufficient in each case, so the COSMO procedure needs to be called, but all optimal solutions are generated very quickly. The pictures contain partial information displayed during the faults analysis.

In case of state-variable filter (Fig. 2 and Table 1), after the sorting phase we obtain the following order of measuring nodes: [7, 5, 6, 2, 1, 4, 3] and the faults are sorted in the following order {2, 7, 13, 9, 11, 4, 12, 10, 14, 5, 8, 1, 3, 6} (these numbers correspond to row numbers of Table 1). Step 6 of the SALTO procedure examines the first (the most significant) node from the list – node #7. This node can uniquely separate 5 faults, i.e. the '*singleton set*'

for this node consists of faults {1, 3, 7, 10 and 11} (the numbers correspond to the rows in Table 1) and these faults are removed from the list of undiagnosed faults. The next node from the list, i.e. node #5 has the '*singleton set*' consisting of faults {1, 3, 10 and 11} (formally ***Single***(n_5) = {S_1, S_3, S_{10}, S_{11}}), so this set is included in the previous. However, nodes #5 and #7 together enable to separate fault 8: ***SEP***({n_5, n_7}, S_8) (with the unique signature (3, 2)).

The SALTO procedure is not able to extend the set of nodes separating the undiagnosed nodes, so after its termination we have still 8 undiagnosed faults (states) - {2, 4, 5, 6, 9, 12, 13 and 14} (see Fig. 5) and the COSMO algorithm should be run. Fortunately, in this case, after step 3a of the procedure the set is extended by three additional nodes #1, #3 and #6, and the set [1, 3, 5, 6, 7] is the optimal one for full separation of the circuit states (faults).

Let us analyze it. Step 2 generates 4 sets of undiagnosed faults, where each set $\boldsymbol{G_k}$ consists of faults with the same signature composed of states of selected nodes #5 and #7, so we have:

$\{n_5, n_7\}_1=(0, 0) \rightarrow \boldsymbol{G}_1=\{2, 4\}$; $\{n_5, n_7\}_2=(0, 2) \rightarrow \boldsymbol{G}_2=\{5, 12\}$;

$\{n_5, n_7\}_3=(2, 0) \rightarrow \boldsymbol{G}_3=\{13, 14\}$ and $\{n_5, n_7\}_4=(3, 0) \rightarrow \boldsymbol{G}_4=\{6, 9\}$.

In step 3, each set is analyzed. Faults 2 and 4 belonging to $\boldsymbol{G}_1$ can be separated only by node #1, so it is selected (Fig. 5). Moreover, this node enables to separate faults 13 and 14 with signatures (states of nodes #1, #5 and #7) equal (0, 2, 0) and (1, 2, 0) respectively. Faults {5, 12} (set $\boldsymbol{G}_2$) and faults {6, 9} (set $\boldsymbol{G}_4$) can be separated only by nodes #6 and #3 respectively. That is why these two nodes are added to the set of selected nodes (Fig. 5).

The second case – the analysis of the leap-frog filter (Fig. 3 and Table 2) is similar, but it requires one more step – step 4 of the COSMO procedure. Fig. 6 presents the listing of the results, which shows that the SALTO algorithm selects four nodes: #4, #6, #10 and #12. This set allows the separation of 11 of circuit states (and 11 states are not recognized yet). The first phase (steps 1-3) of the COSMO algorithm adds to this set two extra nodes: #2 and #8 and we obtain the trivial case. The unresolved set consists only of two states S_2 and {S_9, S_{27}} (lines 3 and 9 in Table 2). Step 4 of the COSMO algorithm suggests 3 additional nodes: #1, #3 or #5. So, in fact we obtain three optimal (7-node) solutions: [1,2,4,6,8,10,12], [2,3,4,6,8,10,12] and [2,4,5,6,8,10,12]

Fig. 7 describes the most complex analysis of the elliptical filter dictionary. In this case, the solution has been obtained after four iterations (step 4 of the algorithm COSMO), where the system has to employ "the heuristic commonsense inference engine" to find the optimal searching strategy and eventually to point the measuring sets (here: [2,3,4,5,8,10,14], [2,3,4,8,10,12,14] and [2,3,4,8,10,13,14]). Table 4 summarizes the results obtained for all filters.

Table 4. Summary of the Results Obtained for Filter Benchmarks.

Circuit name	Dictionary length	File size [B]	Nr of test points	Generation time [ms]
Active filter [15, 19, 20]	19	1426	4	10 (only SALTO)
State-variable filter	14	1274	5	30 (COSMO after step3)
Leapfrog filter	21	2082	7	56 (COSMO 3 iterations)
Elliptical filter	31	2689	7	72 (COSMO after step4)

5. The expert system implementation

System is implemented in LPA Win-PROLOG [24] in MS Windows XP on a Pentium Core Duo (2 GHz) platform. Searching algorithms are implemented as production rules in PROLOG and the backtracking mechanism assures consistency of the entire information within the system. The SALTO algorithm, which separates absolutely necessary measurement nodes, is built only of '*classical logic*' mechanisms offered by PROLOG language (linear resolution). Some steps (especially step 5 of the COSMO) require non-standard heuristic

approaches [25], when the trial-and- error method should be used. The algorithm, presented here, employs heuristic rules based on the Fuzzy Default Logic (FDL) technique introduced in [3].

5.1. Heuristic inference engine based on Fuzzy Default Logic

A detailed analysis of sophisticated logical theories is not the main subject of this paper, so only fundamental definitions used by the commonsense inference engine of the implemented diagnostic expert system are briefly recalled.

Definition* 2 [3]*: The ***Fuzzy Default Rule*** (***FDR***) is defined as the following inference rule:

$$\frac{\alpha : \beta_1, \beta_2 ... \beta_N}{\Phi^{\lambda}}. \tag{4}$$

The above rule could be interpreted in the following way: if $\boldsymbol{\alpha}$ is true, and $\boldsymbol{\beta_1 \ldots \beta_N}$ cannot be proved, infer $\boldsymbol{\Phi^{\lambda}}$ and treat it as a temporary hypothesis (that could be invalidated later). $\boldsymbol{\alpha}$, $\boldsymbol{\beta_1 \ldots \beta_N}$ are *wffs* (well formed formulas) in a given propositional language ***L*** and $\boldsymbol{\Phi^{\lambda}}$ is a ***Fuzzy Hypothesis*** (***FH***) of the following form:

$$\Phi^{\lambda} = \{[h_1^{\lambda}, \mathrm{Tw}(h_1^{\lambda})], [h_2^{\lambda}, \mathrm{Tw}(h_2^{\lambda})], \ldots, [h_m^{\lambda}, \mathrm{Tw}(h_m^{\lambda})]\}, \tag{5}$$

where: $\boldsymbol{h_i^{\lambda}}$ ($\boldsymbol{i}$ = 1...$\boldsymbol{m}$) are wffs in propositional language ***L***, and $\boldsymbol{Tw(h_i^{\lambda})}$ denotes ***Trustworthiness***; i.e. one of the modality of generalized constraints in Zadeh's sense [26] (bivalent, probabilistic, fuzzy, veristic etc.).

Definition* 3 [3]*: The ***Fuzzy Default Logic*** (***FDL***) is the commonsense based theory $\boldsymbol{\Delta_{fuzzy}}$ which divides the inferring process into stages (steps) $\boldsymbol{\Delta^{s}_{fuzzy}}$ and at every step a given hypothesis is generated. The stage $\boldsymbol{\Delta^{s}_{fuzzy}}$ is represented by a quadruple: axioms, simple relations between the knowledgebase elements (classical logic relations), fuzzy default rules and constraints. Formally:

$$\boldsymbol{\Delta_{fuzzy}} = \{ \boldsymbol{\Delta^{s1}_{fuzzy}}, \boldsymbol{\Delta^{s2}_{fuzzy}}, \ldots \boldsymbol{\Delta^{sN}_{fuzzy}} \} \text{ and } \boldsymbol{\Delta^{sk}_{fuzzy}} \{ \boldsymbol{A, Facts, FDRs, C} \} \mapsto \boldsymbol{h_{sk}}. \tag{6}$$

In the presented application (test point selection) hypotheses $\boldsymbol{h_i^{\lambda}}$ consist of nodes (test points' symbols) accompanied with trustworthiness which (depending on the situation) may reflect the priority of a node, the power of an ambiguity set, accessibility of the node etc. The prerequisites $\boldsymbol{\alpha}$ and justifications $\boldsymbol{\beta_1 \ldots \beta_N}$ [25] not always are present (special cases of FDR rules), but usually they reflect the objective state of the analysis or suggestions given during the interaction (see Fig. 1).

It is debatable if such sophisticated tool is really necessary for this purpose, i.e. selection of test nodes. It is true that the proposed extensions of PROLOG backtracking with the additional inference engine complicate the implementation and reduce the efficiency (the system performance), but they add new quality to the system which is now supplied with some '*intelligence*' and new skills. Sometimes it is necessary to introduce non-standard techniques, to solve problems belonging to NP-hard class. The main, unquestionable value of the proposed mechanism is the ability to assume temporal hypotheses (here: nodes' symbols) which may be invalidated later during the deduction process. Another justification that argues for such a solution – the fuzzy default rules are invoked at the very last stages of the searching procedure, and they are used only as additional, supporting tools, to move the search process from a deadlock and to find the further optimal (or semi-optimal) strategy. In case of the presented benchmark examples, these rules are hardly used – in the last example concerning the elliptical filter (Table 3), the system used the rule telling that the remaining faults should

be grouped and the distinguishing nodes should be selected from nodes of the highest priority (the position in the order). Because of that fact node #3 was added in the second iteration. This operation enabled us to separate five faults. Three states: S_{16}, S_{19} and $\{S_{20},S_{22}\}$, described in the rows 12, 17 and 18 of Table 3 respectively, have the signature for nodes #2, #4 and #10 equal to (1, 3, 2). However node #3 allows good separation of those states, with values 2, 3 and 0 respectively. Moreover, adding node #3 to the selected nodes allows to separate two additional circuit states S_{14} and S_{18} (rows 12 and 16 of Table 3). So, in this case the trustworthiness value for the hypothesis node(3) is the highest and it is selected as true. This process of deduction is continued in the subsequent steps of the iteration.

6. Final considerations

To generalize the approach, it should be tested on more complicated dictionaries, unfortunately a uniform library of testbenches of complex practical analog systems does not exist. Some works [9, 22] describe only experiments on randomly generated dictionaries, so this approach has been tested on similar sets of data of 200 dictionaries generated randomly, too. Each dictionary consists of 100 faults and 30 nodes. If the set has only one '*the best*' solution, the approach finds it almost always within the SALTO or COSMO procedure with few iterations, but sometimes (about 2% of cases) we have to run the exclusive procedure for the final sets. These experiments show mathematical properties of the method, but they are far away from professional practice. That is why the results should be interpreted very carefully.

The presented approach gives another contribution to the analog systems testability. In many cases the proposed procedures have shown their benefits – drastically reducing the process of finding the solution. However, we must properly plan the testing experiments and generalize the term '*test node*' [9, 18]. Test nodes should have more levels of representation (AS) and experiments should aid the process of distinguishing errors in characteristic points of the circuit (like outputs) (for example to separate states *FF* and S_3 for the SV-Filter we have to perform a frequency analysis). Another way is to optimize the searching trees and look for better evaluation functions [12] or a dictionary decomposition into subcircuits [20]. The latter techniques allow to deal efficiently with multiple faults. Sometimes it is better to take a less optimal set of points (redundant sets) but a more selective and tolerance-sensitive one.

Finally, we should consider the generalization of the methodology. It is difficult to judge if worse results for more complicated dictionaries disqualify the approach. We must remember that those random dictionaries do not reflect real circuits. So the presented approach is rather a dedicated technique for a specific kind of task than a general-purpose optimization algorithm.

As to computational complexity of the presented algorithms, the most time-consuming phases are sorting procedures, which need (in case of the quick-sort algorithm) $O(S \cdot n \cdot \log(S))$ (where n is the number of nodes and S corresponds to the number of circuit states – faults), so it is the same as in literature [9, 14]. If we assume the worst case, i.e. the employment of both algorithms SALTO and COSMO and a situation when the first algorithm does not give any node (which in fact denotes that the dictionary is not correctly selected), this complexity is not greater than $2 \cdot O(S \cdot n \cdot \log(S))$. However, we have to take into account the fact that both algorithms, even though they look very complicated, perform simple comparisons and any complex evaluation (like counting logarithms in case of the entropy-based approach [14] or genetic iterations [16] are not needed). So when we consider computation complexity, we should also remember the complexity of operations (transformations), which are performed on a real computer with appropriate resources. That is why the obtained results (gathered in Table 4) show the great efficiency of the approach and speed of the evaluation in comparison

with other approaches [9, 22, 27]. Those previous approaches generated results in seconds; SALTO and COSMO produce solutions in milliseconds on the same reference platform (MS Windows). The displays presented in Fig. 6 and 7 give two or more solutions at the same cost, so we can say that we are able to consider optimal results and select the most suitable one in a particular application.

Acknowledgements

The author would like to thank to Prof. Lotfi A. Zadeh for very fruitful discussions and comments during his (author's) stay at UC Berkeley and to Prof. Jerzy Rutkowski, Chair of the Division of Circuit and Signal Theory in the Institute of Electronics, Vice-Rector of Silesian University of Technology, who encouraged him to write this paper.

References

[1] Huertas, I.L. (1993). Test and design for testability of analog and mixed-signal integrated circuits: theoretical basis and pragmatical approaches. *Proceedings of ECCTD'93 Conference*, 75–156.

[2] Pułka, A. (2007). A Heuristic Fault Dictionary Reduction Methodology. *Proceedings of IEEE ICECS 2007 Conference*. Marakesh. MOROCCO, 1115–1118.

[3] Pułka, A. (May 2009). Decision Supporting System Based on Fuzzy Default Reasoning. *Proceedings of the HSI'09 Human Systems Interaction Conference*. Catania. Italy, 32–39.

[4] Kondagunturi, R., Bradley, E., Maggard K., Stroud, C. (1999). Benchmark Circuits for Analog and Mixed-Signal Testing. *Proceedings of IEEE Southeastconf'99*, 217–220.

[5] Lin, P.M., Elcherif, Y.S. (1985). Analogue circuits fault dictionary–New approaches and implementation. *Journal of Circuit Theory Applications*, 13, 149–172.

[6] Hochwald, W., Bastian, J.D. (1979). A dc approach for analog fault dictionary determination. *IEEE Transactions on Circuits and Systems*, 26, 523–529.

[7] Rutkowski, J. (1993). A DC Approach for Analog fault dictionary determination. *Proceedings of ECCTD'93 Conference*, 877–880.

[8] Kamińska, B., Arabi, K., Bell, I., Goteti, P., Huertas, J.L., Kim, B., Rueda, A., Soma, M. (1997). Analog and Mixed-Signal Benchmark Circuits - First Release. *Proceedings of ITC'97*, 183–190.

[9] Golonek, T., Rutkowski, J. (2007). Genetic-Algorithm-Based Method for Optimal Analog Test Points Selection. *IEEE Transactions on Circuits and Systems – II: Express Briefs*, 54(2), 117–121.

[10] Yang, C., Tian, S., Long, B. (2009). Application of Heuristic Graph Search to Test-Point Selection for Analog Fault Dictionary Techniques. *IEEE Transactions on Instrumentation and Measurements*, 58 (7), 2145–2158.

[11] Jiang, R., Wang, H., Tian, S., Long, B. (2010). Multidimensional Fitness Function DPSO Algorithm for Analog Test Point Selection. *IEEE Transactions on Instrumentation and Measurement*, 59(6), 1634–1641.

[12] Farreny, H. (1999). Completeness and Admissibility for General Heuristic Search Algorithms–A Theoretical Study: Basic Concepts and Proofs. *Kluwer Academic Publisher. Journal of Heuristics*, 5(99), 353–376.

[13] Jiang Cui, Youren Wang (2010). A novel approach of analog fault classification using a support vector machines classifier. *Metrology and Measurement Systems*, 17(4), 561–582.

[14] Starzyk, J.A., Nelson, D.E., Sturtz, K. (2000). A mathematical foundation for improved reduct generation in information systems. *Knowledge Informative Systems*, 2(2), 131–146.

[15] Czaja, Z., Zielonko, R. (2003). Fault diagnosis in electronic circuits based on bilinear transformation in 3-D and 4-D spaces. *IEEE Transactions on Instrumentation and Measurements*, 52(1), 97–102.

[16] Bandler, J.W., Salama, A.E. (1981). Fault diagnosis of analog circuits. *Proceedings of IEEE*, 73, 1279–1325.

[17] Jantos, P., Grzechca, D., Rutkowski J. (2009). Fault diagnosis in analog electronic circuits - the SVM approach. *Metrology and Measurement Systems*, 16(4), 583–598.

[18] Toczek, W. (2009). Testing and Diagnostics Strategies for Analog Electronic Circuits, *D.Sc. Thesis*. Gdańsk University of Technology. Gdańsk. (in Polish).

[19] Manetti, S., Piccirilli M., Liberatore, A. (1990). Automatic test point selection for linear analog network fault diagnosis. *Proceedings of IEEE ISCAS'90*, 1, 25–28.

[20] Tadeusiewicz, M., Hałgas, S. (August – September 2008). An efficient method for simulation of multiple catastrophic faults. *Proceedings of IEEE ICECS 2008 Conference*. Malta, 356–359.

[21] Prasad, V.C., Babu, N.S.C. (2000). Selection of test nodes for analog fault diagnosis in dictionary approach. *IEEE Transactions on Instrumentation and Measurement*, 49, 1289–1297.

[22] Starzyk, J.A., Dong Liu, Zhi-Hong Liu, Nelson, D.E., Rutkowski, J. (2004). Entropy-Based Optimum Test Points Selection for Analog Fault Dictionary Techniques. *IEEE Transactions on Instrumentation and Measurements*, 53, 754–761.

[23] Benchmark circuits. *http://www.eng.auburn.edu/~strouce/analogbc/BCoverview.htm* (January 2011).

[24] Logic Programming Associates Ltd. Official Web Site. *http://www.lpa.co.uk/* (January 2011).

[25] Reiter, R. (1980). A logic for default reasoning. *Artificial Intelligence*, 13, 81–132.

[26] Zadeh, L.A. (2008). Is there a need for fuzzy logic?. *Information Sciences*, 178(13), 2751–2779.

[27] Miura, Y., Kato, J. (2006). Fault Diagnosis of Analog Circuits Based on Adaptive Test and Output Characteristics. *Proceedings of the 21st IEEE DFT'06*, 410–418.