BULETINUL INSTITUTULUI POLITEHNIC DIN IAŞI
Publicat de
Universitatea Tehnică „Gheorghe Asachi” din Iaşi
Volumul 66 (70), Numărul 4, 2020
Secţia
ELECTROTEHNICĂ. ENERGETICĂ. ELECTRONICĂ

A STRUCTURAL FAULT COLLAPSING ALGORITHM AND THE APPLICATION OF MACHINE LEARNING TO APPROXIMATE THE NUMBER OF COLLAPSED FAULTS

BY

SABINA-ADRIANA FLORIA[*]

“Gheorghe Asachi” Technical University of Iaşi,
Faculty of Automatic Control and Computer Engineering

Received: October 12, 2020
Accepted for publication: December 28, 2020

Abstract. Fault modelling is a necessary step in circuit testing. A certain fault is observed at the output only in the presence of certain stimuli applied to the circuit inputs. These stimuli are called test vectors. When a full test is performed, some test vectors detect several faults. Therefore, it is not necessary to analyse all faults in the circuit. Identifying a smaller set of faults is called fault collapsing and the development of fault collapsing methods is an important step in testing. In this paper, we present a structural collapsing method to reduce the set of single stuck-at faults for a given combinational circuit. Fault analysis is performed at the gate level, and the proposed method allows the collapsing of equivalent faults in different parts of the circuit due to the used structural collapsing rules. The algorithm has a low complexity and is computationally efficient because the gates in the circuit are analysed in a single pass. In this paper, we also present a brief analysis to approximate the number of collapsed faults using a machine learning technique, namely XGBoost. Experimental results show that XGBoost can learn well the features of a combinational circuit in order to predict the number of collapsed faults, but the prediction performance depends on the number of fan-outs. When datasets contain circuits with a small number of fan-out, the prediction performance is better.

[*]Corresponding author; *e-mail*: sabina.floria@tuiasi.ro

Keywords: fault collapsing; fault dominance; fault equivalence; stuck-at faults; machine learning.

1. Introduction

The manufacture of microelectronics is currently a process with high requirements both in terms of quantity and quality. Regardless of the field in which a particular circuit is used, there is at least one stage of testing it (*e.g.* during the manufacturing phase or during its use). Testing in the manufacturing phase aims to detect any manufacturing defects that may prevent the product from providing quality operation and performance. On the other hand, the test during the operation of the system aims to detect any defect that occurs during the use of the system. In the case of a complex system, with high reliability requirements, one can say that it will be subjected to rigorous and frequent tests to ensure the correct operation of the system. When faults are detected, diagnostic procedures are used to identify and replace faulty components. Thus, when we refer to a certain circuit, testing is of high importance.

This paper addresses the testing of combinational circuits. In the case of these circuits, each of its outputs implements a logic function. We say that there is an error in the functionality of the circuit when at least one of its components does not work according to the specifications. Events that cause an error in the functionality of the circuit are called faults that can be of several types, such as faults in the circuit structure (*i.e.* broken connections, stuck-at faults), substitution type faults etc. An error is detected when an incorrect value is obtained at the output of the circuit. Error detection is possible for certain values provided at the circuit input that allow the observation of a fault at the circuit output. The values that activate and propagate the error to the circuit output are called test vectors.

Fault analysis in a circuit can become more complex when several faults are modelled simultaneously. Modelling an n-order fault involves simultaneously modelling of n distinct faults in the circuit. Faults of order n=1 are called single faults, and the analysis consists in modelling a single fault in the circuit. Otherwise, n-order faults with $n \geq 2$ are called multiple faults, and the analysis is more complex because a fault may influence the behaviour of other faults. For the simplicity of the analysis, this paper addresses modelling with single stuck-at faults.

1.1. Single Stuck-at Faults

The single stuck-at fault model is the most popular approach to fault modelling due to its low complexity. This modelling has three properties (Bushnell and Agrawal, 2000):

- only one connection is faulty at a time;

- a faulty connection is permanently set to logic value 0 (*i.e.* stuck-at-0) or logic value 1 (*i.e.* stuck-at-1);
- the fault can be located at an input or output of a gate.

Due to the fact that a circuit is modelled as an interconnection (*i.e.* netlist) of gates, a stuck-at fault is assumed to affect only the interconnection between gates. The following equation is used to determine the total number of single stuck-at faults in a circuit with *N* connections:

$$Single_{sa-faults} = 2 \cdot N \tag{1}$$

For a qualitative test, it is necessary that all single stuck-at faults are analysed (*i.e.* checking all sources of error). Each single stuck-at fault is detected by a set of test vectors. Based on these sets of test vectors, four possible relations can be defined for any two considered faults: independence, concurrency, dominance and equivalence. These relations are very important in testing, and two of them (*i.e.* equivalence and dominance) can be used to reduce the number of circuit faults.

1.2. Fault Collapsing

There are various methods of fault collapsing addressed in the literature (Adapa *et al.*, 2006; Sandireddy and Agrawal, 2005). The main objective of these methods is to obtain a collapsed fault set as small as possible. Computational resources are also an important issue, especially the processing time. Fault collapsing methods are classified as functional methods and structural methods (Bushnell and Agrawal, 2000). These methods are based on the equivalence and dominance relations between the faults.

Let f_1 and f_2 be two faults, and SV_1 is the test vector set that detects f_1 and SV_2 is the test vector set that detects f_2. The faults f_1 and f_2 are in ***equivalence relation*** if the sets SV_1 and SV_2 contain the same test vectors. When two faults are equivalent, it is sufficient to keep only one of the faults in the analysis, while the quality of the test remains the same. A fault f_1 is said to ***dominate*** a fault f_2 if all the tests that detect fault f_2 also detect fault f_1. In this case, the dominant fault (*i.e.* f_1) can be eliminated.

Functional fault collapsing methods use the logic function of the circuit to identify equivalent or dominant faults. An example of functional fault collapsing is the use of the circuit truth table which must also contain the incorrect output of the circuit for each fault considered. Although such a test is ideal, the processing time is very high and it is impossible to perform the test on large circuits (Bushnell and Agrawal, 2000). Several theorems and a practical algorithm are proposed in (Lioy, 1991) to identify some of the functionally equivalent faults. Tests performed on benchmark circuits show that there are many functionally equivalent faults that can be identified using reasonable computational resources. A graph representation of the relations between faults is used in (Prasad *et al*., 2002) and the transitive closure is applied to identify equivalent faults throughout the circuit. An extension of this method is presented in (Agrawal *et al*., 2003), where dominance fault collapsing is also considered. However, both methods have a high complexity in terms of computational resources.

Structural fault collapsing methods use only the circuit topology and the analysis is performed at the gate level. Therefore, the structural fault collapsing has the advantage of very fast processing, but certain faults are not collapsed because the equivalence and dominance relations cannot be easily identified when the functionality of the entire circuit is considered. A fault folding approach is used in (To, 1973) and is based on a graph representation of the relations between faults, called fault-folded graphs. The circuit is analysed at the gate level, and the gates are processed in an output to input pass. In this iterative process, dominance and equivalence relations are identified using fault-folded graphs. Faults of the fan-out stem and its branches are not considered in the fault folding method. An improvement of the fault folding method is proposed in (Lioy, 1993). In this method, an analysis of the faults on the fan-outs is introduced, and the proposed theorems bring improvements only for certain circuit designs. In (Vimjam and Hsiao, 2006) a generalized concept is used for non-reconvergent fan-outs or reconvergent fan-outs with the same inversion parity and the results are promising on most circuits.

To highlight the difference between structural and functional fault collapsing, we considered the circuit in Fig. 1.

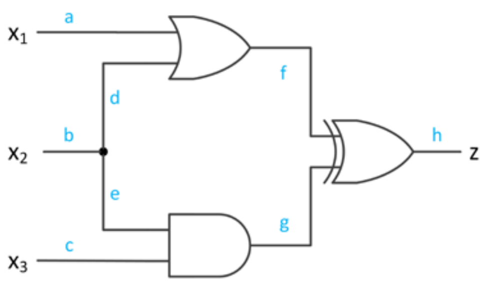


Fig. 1 – Combinational circuit example.

For a complete functional test, all entries in the truth table must be checked. Functional fault collapsing involves computing the logic function for both the correct circuit and all faulty circuits. Note that the circuit has 8 connections, therefore 16 possible faults are considered in Table 1 (*e.g.* a_0 means connection *a* has stuck-at-0 fault). Table 1 describes both the functionality of the correct circuit and the functionality of the faulty circuit (*i.e.* in the presence of a certain fault). In this table, the grey coloured cells represent the incorrect values of the circuit output. Table 2 shows the retained faults after equivalence fault collapsing. For the equivalence collapsed set in Table 2, the functional fault collapsing can be further applied based on the dominance fault collapsing. The retained faults, after dominance fault collapsing, are shown in Table 3.

Table 1
Output of the Correct and Faulty Circuits (for the Circuit of Fig. 1)

a	b	c	z	z – faulty output															
				a_0	a_1	b_0	b_1	c_0	c_1	d_0	d_1	e_0	e_1	f_0	f_1	g_0	g_1	h_0	h_1
0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	1	0	1	0	1
0	0	1	0	0	1	0	0	0	0	0	1	0	1	0	1	0	1	0	1
0	1	0	1	1	1	0	1	1	0	0	1	1	1	0	1	1	0	0	1
0	1	1	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	0	1
1	0	0	1	0	1	1	1	1	1	1	1	1	1	0	1	1	0	0	1
1	0	1	1	0	1	1	0	1	1	1	1	1	0	0	1	1	0	0	1
1	1	0	1	1	1	1	1	1	0	1	1	1	1	0	1	1	0	0	1
1	1	1	0	0	0	1	0	1	0	0	0	1	0	1	0	1	0	0	1

Table 2
Retained Faults After Equivalence Fault Collapsing

a	b	c	z	z – faulty output											
				a_0	a_1	b_0	b_1	c_0	c_1	d_0	e_1	f_0	g_1	h_0	h_1
0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1
0	0	1	0	0	1	0	0	0	0	0	1	0	1	0	1
0	1	0	1	1	1	0	1	1	0	0	1	0	0	0	1
0	1	1	0	0	0	0	0	1	0	1	0	1	0	0	1
1	0	0	1	0	1	1	1	1	1	1	1	0	0	0	1
1	0	1	1	0	1	1	0	1	1	1	0	0	0	0	1
1	1	0	1	1	1	1	1	1	0	1	1	0	0	0	1
1	1	1	0	0	0	1	0	1	0	0	0	1	0	0	1

Table 3
Retained Faults After Dominance Fault Collapsing

a	b	c	z	z – faulty output							
				a_0	a_1	b_0	b_1	c_0	c_1	d_0	e_1
0	0	0	0	0	1	0	1	0	0	0	0
0	0	1	0	0	1	0	0	0	0	0	1
0	1	0	1	1	1	0	1	1	0	0	1
0	1	1	0	0	0	0	0	1	0	1	0
1	0	0	1	0	1	1	1	1	1	1	1
1	0	1	1	0	1	1	0	1	1	1	0
1	1	0	1	1	1	1	1	1	0	1	1
1	1	1	0	0	0	1	0	1	0	0	0

In structural fault collapsing, the relations between faults are identified at the gate level. A basic structural fault collapsing involves the analysis of each gate in the circuit, where faults are locally collapsed. For example, Fig. 2 illustrates the equivalence ($"\leftrightarrow"$) and dominance ($"a_1 \rightarrow b_1"$ means a_1 dominates b_1) relations between faults for the AND gate.

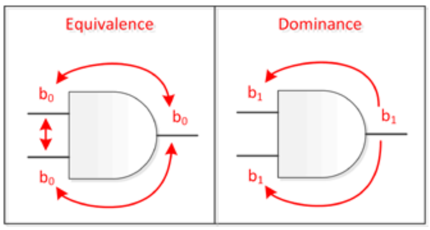


Fig. 2 –The relations between faults for the AND gate.

All stuck-at-0 faults for the AND gate are equivalent, while the stuck-at-1 fault on the output dominates all stuck-at-1 faults on the inputs. Another approach to show these relations is based on the AND function, where we consider both the correct output and the faulty one (*i.e.* in the presence of certain faults) (Table 4). Note that in Table 4 the faults a_0, b_0, c_0 are equivalent because they are detected by the same test vector (*i.e.* ab = 11). In this case, two faults can be removed from the analysis. Also, note that faults a_1 and b_1 are detected by test vectors ab=01 and ab=10. However, the test vectors ab=01 and ab=10 are included in the test vectors set of the fault c_1. Therefore, the faults a_1 and b_1 are dominated by c_1, and c_1 (*i.e.* the dominant fault) can be eliminated.

Table 4
The AND Function with the Correct Output and the Faulty Output

a	*b*	*z*	*z* – faulty output					
			a_0	a_1	b_0	b_1	c_0	c_1
0	0	0	0	0	0	0	0	1
0	1	0	0	1	0	0	0	1
1	0	0	0	0	0	1	0	1
1	1	1	0	1	0	1	0	1

Fault analysis at the gate level brings a distinct advantage in the process of fault collapsing for the entire circuit. The advantage is a short processing time because each gate in the circuit is processed only once. When a gate is analysed, the faults that can be eliminated should be identified. Note that in Table 4 the dominant faults can always be eliminated. In the case of equivalent faults, any of them may be chosen to be eliminated. However, at least one equivalent fault must be retained as a representative. Depending on which representative equivalent fault is chosen and the order in which the gates are processed, the structural fault collapsing can provide different results.

In this paper, (*i*) we propose a low complexity structural fault collapsing algorithm, where processing is done only at the gate level and (*ii*) we use a machine learning technique in order to approximate the number of collapsed faults. The purpose of this brief analysis is to observe the performance of such a technique when only the circuit structure is provided as input.

2. Model Description

In the proposed method, we consider two main aspects: the gates processing order and the fault collapsing procedure at a specific gate.

The gates processing order is important in the structural fault collapsing because the final outcome (*i.e.* the collapsed faults) may be different in some cases. However, the input to output pass produces a unique result since a gate has a single output and no collapsing is possible through fan-outs (Bushnell and Agrawal, 2000). Therefore, we choose the input to output pass to obtain a unique result for the number of collapsed faults.

When analysing a gate, it is also important how the faults are collapsed. In the proposed method, we are using both dominance and equivalence fault collapsing at each processed gate. In the case of dominance fault collapsing, the dominant fault at a gate is always eliminated (*i.e.* the fault on the output). However, in the case of equivalence fault collapsing, it is important which representative fault is chosen. The proposed method is inspired from (Bushnell and Agrawal, 2000) because we always eliminate all equivalent faults on the gate inputs and keep the fault on the output as a representative. The choice of this representative fault is justified by the gates processing order, (*i.e.* the input to output pass) which is helpful because the representative fault could be eliminated when analysing a gate at the next level.

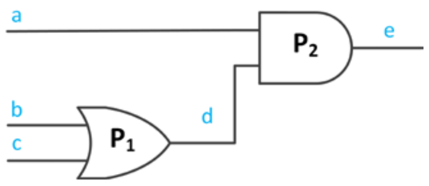


Fig. 3 – Case study of a combinational circuit.

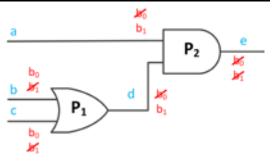
Our contribution is described by the following case study: let be two gates P_1 and P_2 which are interconnected as illustrated in Fig. 3. Since we are using an input to output pass, gate P_1 is analysed before gate P_2. The fault d_0 is the dominant for OR gate P_1, therefore this fault is eliminated. When gate P_2 is analysed, note that a_0, d_0 and e_0 are equivalent faults (AND gate). Since d_0 was previously eliminated, faults a_0 and e_0 can also be eliminated. Using this rule, the structural fault collapsing is improved due to a propagation effect in which faults from different gates can be collapsed.

To exemplify the proposed method, we summarize the rules that we use for the structural fault collapsing process:

R1) the gates are processed in a single input to output pass;
R2) a gate can only be processed if all its inputs come from other gates already processed;
R3) the dominant fault (at the gate output) is always eliminated;
R4) if none of the equivalent faults have been previously eliminated, the equivalent faults on the gate inputs are eliminated and the fault on the output is retained as representative;
R5) if at least one equivalent fault was previously removed, then all equivalent faults are removed.

The rule R5 is introduced in this paper in addition to R1 – R4 from (Bushnell and Agrawal, 2000). Each gate is processed only once (fast method), and the processing of a gate consists in the application of rules R3 – R5. Table 5 shows the use of these rules for the circuit in Fig. 3.

Table 5
Structural Fault Collapsing Using the Proposed Method. Equivalent Faults are Displayed in Bold

Circuit	Gate	Removed faults	Notes
 a $\cancel{b_0}$ b_1 P_2 e $\cancel{b_0}$ $\cancel{b_1}$ b_0 $\cancel{b_1}$ b c P_1 d $\cancel{b_0}$ b_1 b_0 $\cancel{b_1}$ *Number of collapsed faults: 4*	*I. P_1*	$\mathbf{b_1}$, $\mathbf{c_1}$, d_0	*Rule R4): remove b_1, c_1 and keep d_1* *Rule R3): remove d_0*
	II. P_2	$\mathbf{a_0}$, $\mathbf{e_0}$, e_1	*Rule R5): d_0 was removed, remove all equivalent faults (a_0, e_0)* *Rule R3): remove e_1*

2.1. Implementation

The structural fault collapsing uses only the circuit topology. Therefore, it is first necessary to model the circuit structure. In the proposed method, we model the circuit as a directed graph in which the vertices are represented by the gates and the edges are the interconnections between gates. We consider vertices and edges as entities with different properties. To facilitate the processing of graph entities, we propose some ideas for modelling the given circuit. First, the primary inputs and outputs of the circuit are a special case because these connections have a missing vertex. For example, a primary input is a directed edge that is incident on a vertex, but does not come from a vertex.

To overcome this problem, we introduce special gates to represent the primary inputs and outputs of the circuit. These special gates are entities that do not influence the circuit functionality. To illustrate the modelling with the aforementioned conventions, we use as an example the circuit in Fig. 4 with the associated graph illustrated in Fig. 5*a*. In this modelling, we also take into account that the fan-outs (*i.e.* stem and branches) are connections with independent faults. The faults of these connections are considered independent because fault collapsing is not possible on the fan-outs (Bushnell and Agrawal, 2000). Therefore, the graph in Fig. 5*a* is not a sufficient representation because the fan-out stem is not modelled. To overcome this problem, we introduce an additional special gate with a single input and multiple outputs. The input of the new gate represents a fan-out stem, while the outputs are correlated with the fan-out branches. We illustrate in Fig. 5*b* the special gate F_1 which models all the fan-outs.

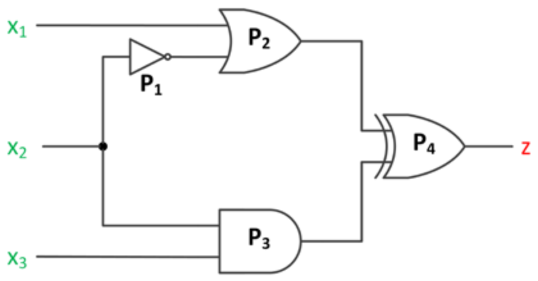


Fig. 4 – Combinational circuit example.

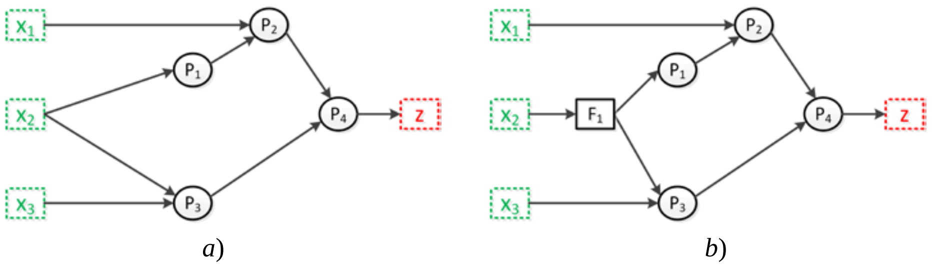


Fig. 5 – Modelling the circuit as a directed graph: *a*) the fan-out stem is not modelled, *b*) the fan-out stem is modelled using the special gate F_1.

After modelling the circuit as a directed graph, we assign a fault list for each connection (*i.e.* edges). Fault lists are initialized with the two possible faults: stuck-at-0 and stuck-at-1. Let c_{ij} be the connection between two gates i and j. The faults of the connection c_{ij} are stored in a list denoted L_{ij} which is used in the adjacency matrix. If two vertices i and j are connected (directed connection), then they have an associated list L_{ij} in the adjacency matrix. Otherwise, two unconnected vertices will not have an associated list. In this way, we model both the circuit structure and the faults of each connection. Considering the graph from Fig. 5*b*, its adjacency matrix (M) is:

$$M = \begin{array}{c|ccccccccc|} & x_1 & x_2 & x_3 & F_1 & P_1 & P_2 & P_3 & P_4 & z \\ \hline x_1 & - & - & - & - & - & L_{1,6} & - & - & - \\ x_2 & - & - & - & L_{2,4} & - & - & - & - & - \\ x_3 & - & - & - & - & - & - & L_{3,7} & - & - \\ F_1 & - & - & - & - & L_{4,5} & - & L_{4,7} & - & - \\ P_1 & - & - & - & - & - & L_{5,6} & - & - & - \\ P_2 & - & - & - & - & - & - & - & L_{6,8} & - \\ P_3 & - & - & - & - & - & - & - & L_{7,8} & - \\ P_4 & - & - & - & - & - & - & - & - & L_{8,9} \\ z & - & - & - & - & - & - & - & - & - \end{array} \quad (2)$$

Graph vertices are entities with two main properties: the vertex type (*i.e.* the gate type) and a flag that indicates whether the vertex has been processed or not. The vertex type is used to apply specific fault collapsing rules for each gate type. The structural fault collapsing is not possible at the following gates: the XOR gate and the special gates (*i.e.* input, output and fan-out gates). In the initialization step, we mark only the special input gates as processed gates. Then, the input to output pass is performed by identifying other viable gates that can be processed. A gate is viable when all its inputs come only from other processed gates. The structural fault collapsing procedure has a low complexity and is described in the Pseudocode 1.

Pseudocode 1. The structural fault collapsing procedure

```
while unprocessed nodes exists do
   node = GetViableUnprocessedNode()
   for each input Cin of node do
      F = GetEquivalenceClassFault(Cin)
      if fault F is reduced then
         ReduceAllEquivalentFaults(node, F)
         break
      else
         ReduceFault(Cin, F)
      end
   end
   ReduceDominanceFault(node)
   MarkProcessedNode(node)
end
```

2.2. Estimation of the Number of Collapsed Faults

The main limitation of the structural fault collapsing methods is that the dominance or equivalence relations cannot be identified for some faults at the level of the entire circuit. Functional fault collapsing can identify all equivalent or dominant faults, but it has the disadvantage that the processing time is high. Approximate fault collapsing methods (Al-Asaad and Lee, 2002) could be a compromise solution between functional and structural fault collapsing.

In this paper, we investigate the possibility of using a machine learning technique to approximate the number of collapsed faults of a given circuit. A machine learning technique can be trained to detect certain features from the datasets provided. In our problem, we propose that a dataset contain several circuits with the same structure, but with different gates that are randomly generated. In the generated datasets, we include for each circuit the number of collapsed faults that must be learned by the machine learning method. The number of collapsed faults is obtained with the proposed structural fault collapsing method. A circuit in the dataset is described by the following information:

- the corresponding adjacency matrix to the circuit graph (the matrix contains Boolean values where a value of 1 denotes a connection between two nodes or 0 otherwise);
- gate types (an integer value is associated for each gate, Table 6);
- the number of collapsed faults to be learned.

To generate a dataset, we classify the gates of a circuit into three main categories, according to Table 6. Special gates belong to category I and these are the only gates that are not randomly generated when we create a new circuit. Category II contains gates with a single input, while category III contains multi-input gates.

Table 6
Coding of the Gates

Category	Gate	Code	Notes
I	*Input*	1	*Special gates: they are not randomly generated from one circuit to another*
	Output	2	
	Fan-out	3	
II	*Buff*	4	*Single-input gates*
	NOT	5	
III	*AND*	6	*Multi-input gates*
	NAND	7	
	OR	8	
	NOR	9	
	XOR	10	
	XNOR	11	

When we create a new circuit, we randomly generate the gates belonging to categories II and III, while the circuit topology is maintained. To ensure compatibility between gates, one gate can only be replaced by another gate in the same category. For example, a single-input gate (*i.e.* category II) cannot be replaced by a two-input gate (*i.e.* category III) because they are not compatible.

Fig. 6 illustrates how a combinational circuit is described in a dataset. The circuit is described by the adjacency matrix, the gate types and the number of collapsed faults. The integer associated with a gate is shown in brackets on the circuit graph. The complete description of the circuit is written on a single line. Therefore, the adjacency matrix is written as a vector.

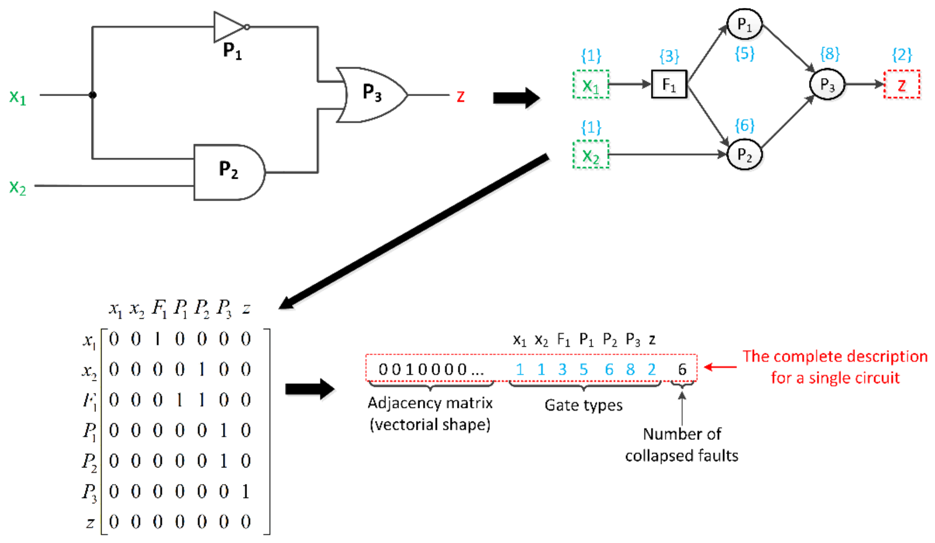


Fig. 6 – Description of a combinational circuit in a dataset.

A machine learning technique consists of the training and testing phases, respectively. In the training phase, the model tries to make a correlation between the features of the circuit (*i.e.* the adjacency matrix and the gate types) and the number of collapsed faults. This phase uses a large number of circuits in the dataset, while the rest of the circuits are used for the testing phase. The purpose of the testing phase is to evaluate the prediction performance of the model when unknown circuits are provided. For each dataset 90% of the circuits are used for training, while the remaining 10% are used for testing.

In this paper, we choose XGBoost (Chen and Guestrin, 2016) to approximate the number of collapsed faults in a combinational circuit. The choice of method is justified based on the results of our previous work (Floria *et al.*, 2019) when we observed that XGBoost has a shorter processing time than other methods, such as autoencoders or convolutional neural networks. The base classifiers of the XGBoost method are the decision trees. When a single decision tree is used, the classification is prone to overfitting. However, using multiple decision trees in an ensemble can lead to better results. Boosting decision trees is an ensemble technique in which several decision trees are used iteratively to improve prediction performance. XGBoost benefits from improved prediction performance and processing time because it includes a sparsity aware algorithm for an efficient processing of datasets with lots of zero values (*e.g.* a dataset with adjacency matrices). Thus, due to the sparsity aware algorithm, only the relevant elements are processed, achieving a significant improvement of the processing time.

We use XGBoost on the regression task and we vary the value of two parameters: the number of boosted trees and the maximum tree depth. Table 7 shows several configurations with different values for the two parameters and we note these configurations with XGB_n (n=1, ..., 7).

Table 7
XGBoost Configurations

Configuration	XGB_1	XGB_2	XGB_3	XGB_4	XGB_5	XGB_6	XGB_7
Maximum tree depth	10	5	5	5	3	3	3
Number of boosted trees	150	150	200	250	150	200	250

Although the number of collapsed faults does not help to identify specific faults in the circuit, the purpose of this brief investigation is to observe whether a machine learning technique can effectively predict the number of collapsed faults in a combinational circuit.

3. Results

This section first presents the results of the structural fault collapsing method on several ISCAS'85 benchmark circuits. We show in Table 8 the results obtained with the proposed method, as well as with the following methods: the fault folding method (To, 1973), the graph method (Prasad *et al*., 2002) and a method based on binary decision diagrams (Ubar *et al*., 2015).

The proposed method provides better results than the fault folding method (To, 1973) and no analysis with fault-folded graphs is required. Note that the graph method (Prasad *et al*., 2002) provides the same results as the proposed method. Also, the graph method is computationally expensive due to the transitive closure performed on graphs, while the proposed method processes the gates in a single input to output pass (*i.e.* a very short processing time). The results of the method based on binary decision diagrams (Ubar *et al*., 2015) are superior, but it is more elaborate than the proposed method.

Table 8
Number of Collapsed Faults for ISCAS’85 Benchmark Circuits: Comparison with Other Methods

Circuit	Number of faults				
	Total	(To, 1973)	(Prasad *et al.*, 2002)	(Ubar *et al.*, 2015)	The proposed method
*c*432	864	458	449	-	449
*c*499	998	730	706	-	706
*c*880	1760	763	-	-	745
*c*1355	2710	1234	1210	1210	1210
*c*1908	3816	1568	1566	1243	1566
*c*2670	5340	2324	2317	1989	2317
*c*3540	7080	2882	2786	2340	2786
*c*5315	10630	4530	4492	3900	4492
*c*6288	12576	5840	5824	5824	5824
*c*7552	15104	6163	6132	5156	6132

3.1. Case Studies for Estimating the Number of Collapsed Faults

To evaluate the performance of XGBoost configurations (Table 7), we generate different datasets with combinational circuits. All circuits in a specific dataset have the same topology, but from one circuit to another the gate types are randomly generated. We choose two main topologies for the generated circuits: *c*432 circuit topology and *c*499 circuit topology (ISCAS'85 benchmark circuits). The structural information of these two circuits is shown in Table 9. Note that both circuits have a similar size (*i.e*. the relative change between *c*432 and *c*499 is 26.25%). However, the main difference between *c*432 and *c*499 circuits is the average number of fan-outs (*i.e*. the relative change is 63.77%). The average number of fan-ins is the ratio between the number of inputs at each gate and the total number of gates. The average number of fan-outs is the ratio between the number of fan-out branches and the total number of fan-out stems.

Table 9
Structural Information of c432 *and* c499 *Benchmark Circuits*

Circuit	Inputs	Outputs	No. gates	Average number of fan-ins	Average number of fan-outs
*c*432	36	7	160	2.10	2.65
*c*499	41	32	202	2.02	4.34

We also investigate the prediction performance of XGBoost configurations when using datasets of different sizes. Table 10 describes the characteristics of each dataset and we choose suggestive names for them (for example, *c*432-3000 is a dataset that contains 3000 circuits, each with the *c*432 circuit topology).

Table 10
Description of Datasets

Dataset name	No. generated circuits	Notes
*c*432-1000	1000	*Each generated circuit has the c432 circuit topology*
*c*432-3000	3000	
*c*499-1000	1000	*Each generated circuit has the c499 circuit topology*
*c*499-3000	3000	

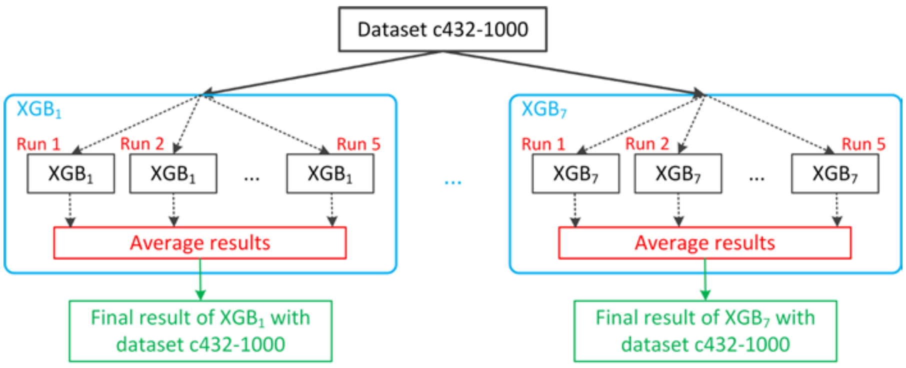
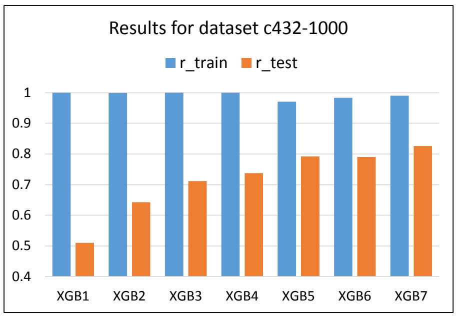


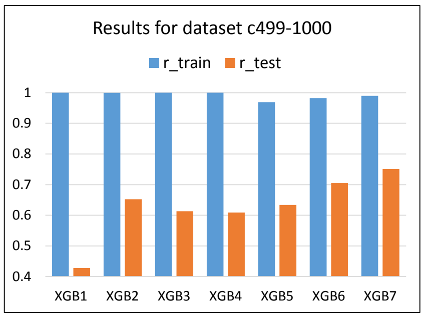
Fig. 7 – Providing a dataset on different XGBoost configurations.

To predict the number of collapsed faults, a dataset is provided for only one XGBoost configuration at a time. We use all the proposed configurations on each dataset to compare the prediction performance. Moreover, we use the average of 5 different runs for each configuration to improve the accuracy of the results. Fig. 7 illustrates how the final results are obtained for a single dataset using each XGBoost configuration.

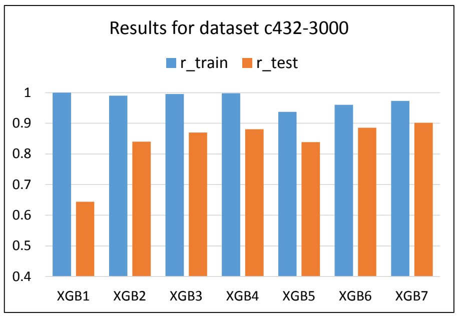
The experimental results for each dataset and XGBoost configuration are illustrated in Fig. 8. The chosen configurations are illustrated on the *X* axis, while the prediction performance is illustrated on the *Y* axis. The prediction performance of each configuration is obtained using the correlation coefficient for both the training phase (r_{train}) and the testing phase (r_{test}). The correlation coefficient is a similarity measure between the desired outputs (*i.e.* the provided number of collapsed faults) and the predicted outputs of the XGBoost configurations. When the predictions of an XGBoost configuration are closer to the expected values, the correlation coefficient is closer to 1. For a better view of the results, the lower boundary of the *Y* axis is set to 0.4, *i.e.* a value close to the performance of the weakest model (XGB_1 in Fig. 8*b*).



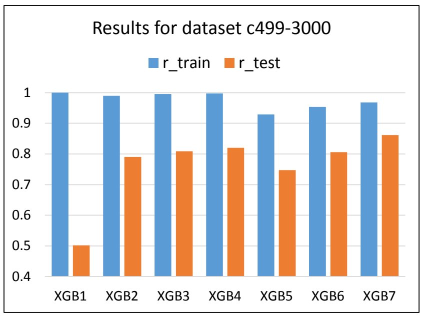
a)



b)



c)



d)

Fig. 8 – Prediction performance of XGBoost configurations for datasets with: *a*) - *b*) 1000 circuits, *c*) - *d*) 3000 circuits.

Fig. 8 shows that XGBoost has a good prediction performance in the training phase (*i.e.* r_{train} has a value greater than 0.9 in all case studies). Therefore, all configurations learn well the features of the circuits in the training phase.

In the sequel, we discuss the prediction performance of the testing phase (*i.e.* how well XGBoost can generalize). Comparing the prediction performance of XGBoost configurations on datasets with 1000 circuits (Fig. 8*a* and Fig. 8*b*), the model configurations have a lower generalization performance (r_{test}) when the circuits have a larger number of fan-outs. The performance difference can be observed by comparing the average difference of the values for XGB_1-XGB_7. In the case of datasets with 1000 circuits, the average prediction accuracy is 0.715 (Fig. 8*a*) for circuit *c*432 and 0.627 (Fig. 8*b*) for circuit *c*499. Thus, the average difference for datasets with 1000 circuits is 0.088. Similarly, the average prediction accuracy for 3000-circuit datasets is 0.837 (Fig. 8*c*) and 0.762 (Fig. 8*d*), with an average difference of 0.075. This observation indicates that predicting the number of collapsed faults is more difficult when the circuits have a larger number of fan-outs. For such circuits, larger datasets are recommended. Fig. 8 shows that the results are worse for XGB_1. One possible reason for this performance is that deeper trees are better predictors on the training set (*i.e.* overfitting), limiting the ability to make good predictions on the testing set.

Based on the obtained results, we can say that XGBoost can learn well the features of combinational circuits that contain a smaller number of fan-outs. If the circuits have a larger number of fan-outs, datasets with a larger number of circuits must be used.

4. Conclusions

In this paper, we propose a structural fault collapsing method that can eliminate certain equivalent faults in different parts of the circuit. The limitation of the method is given by fan-outs and XOR gates, where the structural fault collapsing is not possible. The algorithm has a low complexity and is applied on several ISCAS'85 benchmark circuits. After obtaining the number of collapsed faults with the proposed method, we used XGBoost to approximate the number of collapsed faults of a circuit. The tests are performed on a circuit with a smaller number of fan-outs (*c*432) and on a circuit with a higher number of fan-outs (*c*499). Experimental results show that the average prediction accuracy for 1000 circuits is 0.715 (c432) and 0.627 (c499), while for 3000 circuits it is 0.837 (c432) and 0.762 (c499), implying that XGBoost can have a better prediction performance when circuits include a smaller number of fan-outs.

The proposed structural fault collapsing method eliminates equivalent faults at the gate inputs. Since the gates are processed in a single input to output pass, some faults of the fan-out stems are retained. A future direction of research would be to include a complex analysis of fan-out faults to improve fault collapsing.

REFERENCES

Adapa R., Tragoudas S., Michael M.K., *Evaluation of Collapsing Methods for Fault Diagnosis*, Proceedings of the 7th International Symposium on Quality Electronic Design, March 2006, San Jose, CA, USA, 439-444.

Agrawal V.D., Prasad A.V.S.S., Atre M.V., *Fault Collapsing via Functional Dominance*, Proceedings International Test Conference 2003, October 2003, IEEE, Charlotte, NC, USA, **1**, 274-280.

Al-Asaad H., Lee R., *Simulation Based Approximate Global Fault Collapsing*, Proc. International Conf. on VLSI, 2002, 72-77.

Bushnell M.L., Agrawal V.D., *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*, Springer, Boston, MA, 690, 2000.

Chen T., Guestrin C., *XGBoost*: *A Scalable Tree Boosting System*, Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 2016, Association for Computing Machinery, New York, NY, United States, 785-794.

Floria S.-A., Leon F., Caşcaval P., Logofătu D., *An Evaluation of Various Regression Models for the Prediction of Two-Terminal Network Reliability*, 28th

International Conference on Artificial Neural Networks, September 2019, Springer, Cham, Munich, Germany, 267-280.

Lioy A., *Looking for Functional Fault Equivalence*, IEEE Proceedings International Test Conference 1991, October 1991, IEEE, Nashville, TN, USA, 858-864.

Lioy A., *On the Equivalence of Fanout-Point Faults*, IEEE Transactions on Computers, **42**, 3, 268-271 (1993).

Prasad A.V.S.S., Agrawal V.D., Atre M.V., *A New Algorithm for Global Fault Collapsing into Equivalence and Dominance Sets*, Proceedings International Test Conference 2002, October 2002, IEEE, Baltimore, MD, USA, 391-397.

Sandireddy R.K.K.R, Agrawal V.D., *Diagnostic and Detection Fault Collapsing for Multiple Output Circuits*, Proceedings of the conference on Design, Automation and Test in Europe, DATE'05, Mar 2005, Munich, Germany, **5**, 1014-1019.

To K., *Fault Folding for Irredundant and Redundant Combinational Circuits*, IEEE Transactions on Computers, **C-22**, 11, 1008-1015 (1973).

Ubar R., Jürimägi L., Orasson E., Raik J., *Fault Collapsing in Digital Circuits Using Fast Fault Dominance and Equivalence Analysis with SSBDDs*, 23rd IFIP WG 10.5/IEEE International Conference on Very Large Scale Integration, VLSI-SoC 2015, October 2015, Springer, Cham, Daejeon, Korea, 23-45.

Vimjam V.C., Hsiao M.S., *Efficient Fault Collapsing via Generalized Dominance Relations*, 24th IEEE VLSI Test Symposium, May 2006, IEEE, Berkeley, CA, USA, 265-270.

UN ALGORITM DE REDUCERE STRUCTURALĂ A DEFECTELOR ȘI APLICAREA ÎNVĂȚĂRII AUTOMATE PENTRU APROXIMAREA NUMĂRULUI REDUS DE DEFECTE

(Rezumat)

Modelarea defectelor este o etapă necesară în testarea circuitelor. Un anumit defect se manifestă doar în prezența anumitor stimuli aplicați la intrările circuitului. Acești stimuli se numesc vectori de test. Atunci când se realizează o testare completă, unii vectori de test detectează mai multe defecte. Prin urmare, nu este necesară analiza tuturor defectelor din circuit. Identificarea unui set redus de defecte se numește reducerea defectelor, iar dezvoltarea unor metode de reducere a defectelor este un pas important în testare. În acest articol se prezintă o metodă de reducere structurală care are în vedere reducerea setului de defecte singulare de tip blocaj pentru un circuit logic combinațional. Analiza defectelor este realizată la nivel de poartă logică, iar metoda propusă permite reducerea defectelor echivalente din regiuni diferite ale circuitului datorită regulilor de reducere structurală utilizate. Algoritmul are o complexitate redusă și acesta este eficient din punct de vedere computațional deoarece porțile logice din circuit sunt analizate printr-o singură parcurgere. De asemenea, în acest articol se prezintă și o analiză succintă în ceea ce privește estimarea numărului redus de defecte folosind o tehnică de învățare automată, și anume XGBoost. Rezultatele experimentale indică faptul că XGBoost poate învăța în mod eficient trăsăturile unui circuit logic combinațional pentru a prezice numărul redus de defecte, dar performanța de predicție depinde de numărul conexiunilor de fan-out. Atunci când seturile de date conțin circuite cu un număr mai mic de conexiuni de fan-out, performanța de predicție este mai bună.