COE-571 Digital System Testing

A Pattern Ordering Algorithm for Reducing the Size of Fault Dictionaries

Authors:
P. Bernardi, M. Grosso, M. Rebaudengo,
M. Sonza Reorda

Presented By:
Farhan Khan

Outline of Presentation

- Abstract
- Introduction
- Tree-based Dictionary Organization
- Dictionary Minimization
  - Dropping Procedure
  - Ordering Procedure
- Experimental Results
- Conclusion
- References

Abstract

- This paper proposes a novel technique for reducing fault dictionary size for combinational and scanned circuits by means of pattern ordering.
- The proposed algorithm manipulates conventional tree-based fault dictionaries.
- In tree-based structures, faults are diagnosed by traversing the tree from its root to a leaf.
- The aim is to globally reduce the length of such paths by a modified patterns order, thus also reducing the dictionary size.

Introduction

- Fault diagnosis processes are computationally hard and memory expensive.
- The structures which are used to store all the information needed to individuate a fault are called *fault dictionaries*.
- Many dictionary organizations have been proposed in literature using matrices, tables, lists and trees.
- The quality of a dictionary organization is given by the *diagnostic expectation* they afford, which is a measure of average size of undistinguished fault classes over all faults.
- The proposed technique is based on a tree-based fault dictionary representation.

Introduction

- The main contribution of this paper is an algorithm able to manipulate such a tree-based fault dictionary and resulting in more effective sequences of patterns.
- In a tree-based fault dictionary, an equivalent class is individuated by traversing the diagnostic tree from its root to a leaf.
- The goal is the minimization of these paths by means of pattern reordering.
- The advantage of using such ordered patterns is twofold:
  - It minimizes the number of information in the fault dictionary, and hence its size.
  - It reduces the average duration of the diagnostic process.

Tree-based dictionary organization

- The tree-based representation follows the principles of generating compact dictionaries.
- It requires two consecutive construction steps:
  - **Step I:** a pass/fail binary tree is built to preliminarily group faults in equivalent classes determined using detection information only.
  - **Step II:** an output-based tree is built for each of those classes coming from step I, to further distinguish within their faults by observing faulty circuit responses.

A TEST SET FOR *c17*

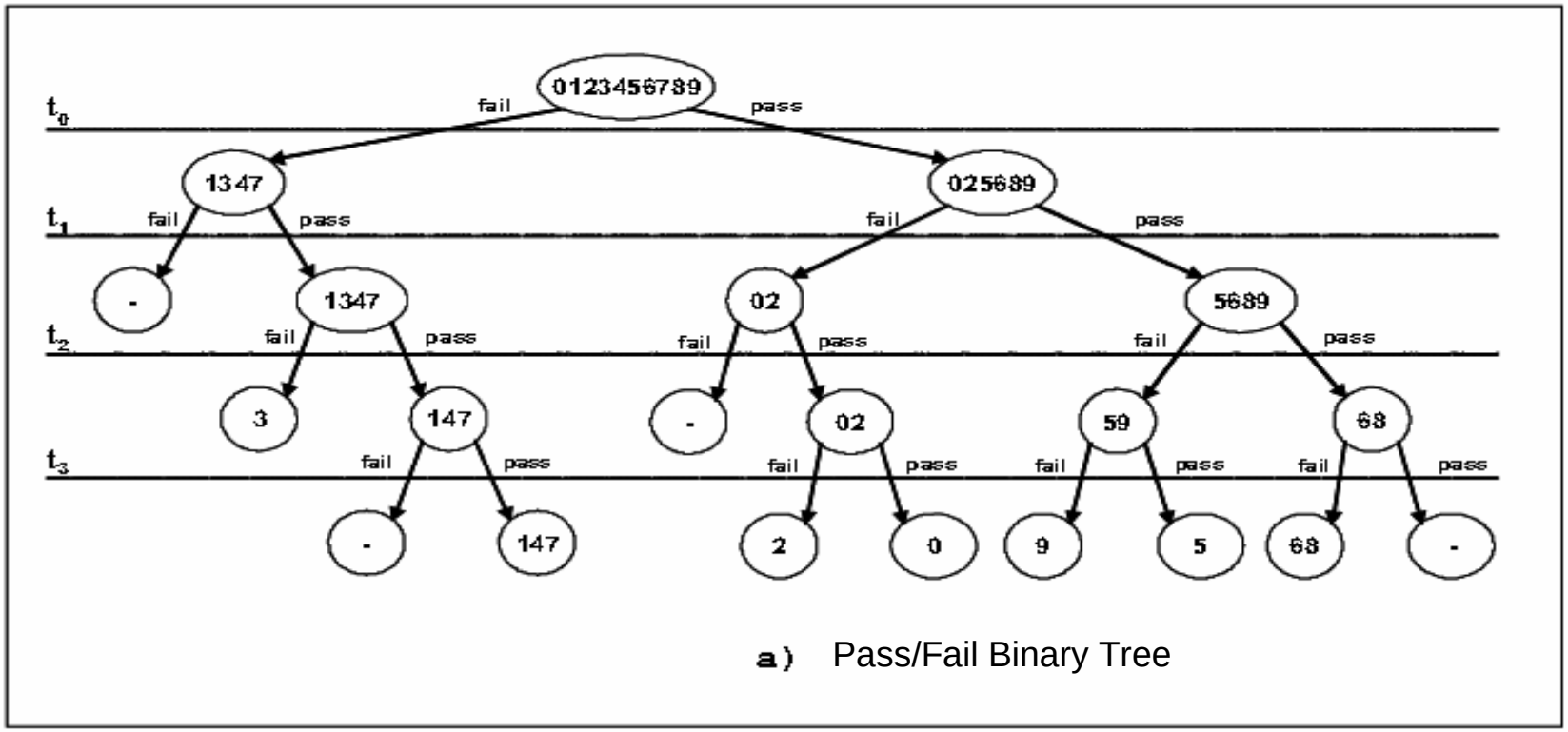
	tests	fault-free response
0	10010	00
1	01111	00
2	11010	11
3	10101	11

TABLE 2
A TWO-DIMENSIONAL FULL RESPONSES DICTIONARY

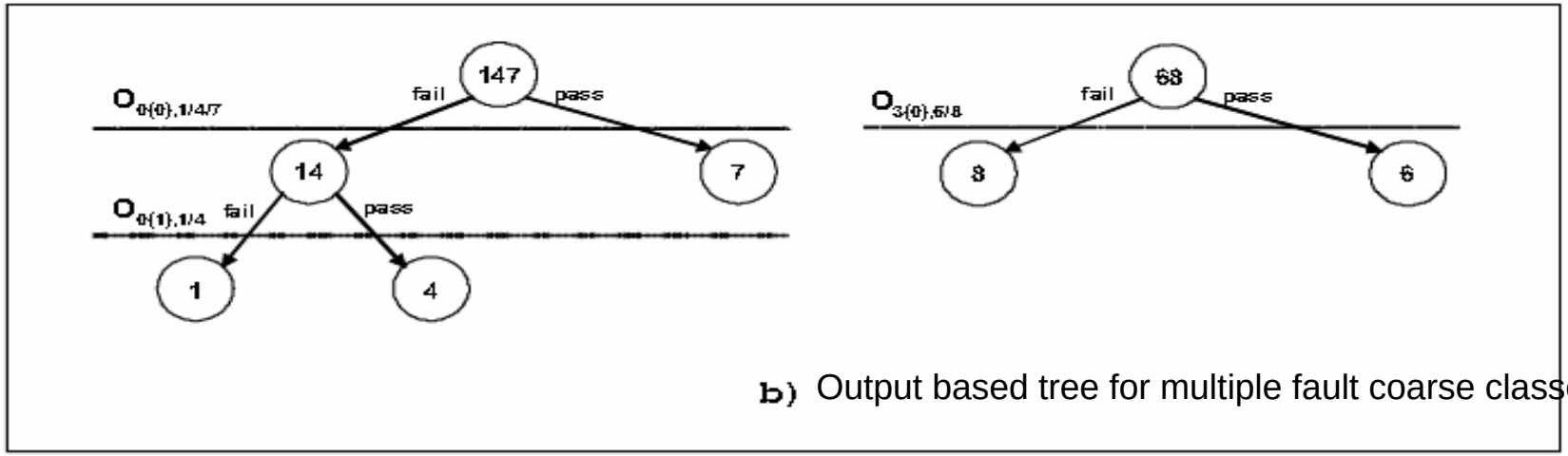
	response			
fault	0	1	2	3
0	00	10	11	11
1	11	00	11	11
2	00	11	11	01
3	10	00	10	11
4	10	00	11	11
5	00	00	00	11
6	00	00	11	10
7	01	00	11	11
8	00	00	11	01
9	00	00	00	10

TABLE 3
A TWO-DIMENSIONAL PASS/FAIL DICTIONARY

	response			
fault	0	1	2	3
0	0	1	0	0
1	1	0	0	0
2	0	1	0	1
3	1	0	1	0
4	1	0	0	0
5	0	0	1	0
6	0	0	0	1
7	1	0	0	0
8	0	0	0	1
9	0	0	1	1



a) Pass/Fail Binary Tree



b) Output based tree for multiple fault coarse classes

Tree-based dictionary organization

- Three tables are required to store the obtained fault dictionary:
- **Coarse classes table:** it associates a number to a set of faults, whose equivalence is determined using the pass/fail binary tree.
- **Pass/Fail table:** it stores the pass/fail sequence for each class included in the coarse classes table.
- **Fine classes table:** for each coarse class ccj in the CC* set of coarse classes including more than 1 fault, it itemizes the fine classes isolated by building output-based trees, this information is stored resorting to a list-like representation.

TABLE I: Coarse classes table.

Coarse class	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Equivalent faults	3	1:4:7	2	0	9	5	6:8

TABLE II: Pass/Fail table.

Coarse class	t_0	t_1	t_2	t_3
0	1	0	1	
1	1	0	0	0
2	0	1	0	1
3	0	1	0	0
4	0	0	1	1
5	0	0	1	0
6	0	0	0	1

TABLE III: Fine classes table.

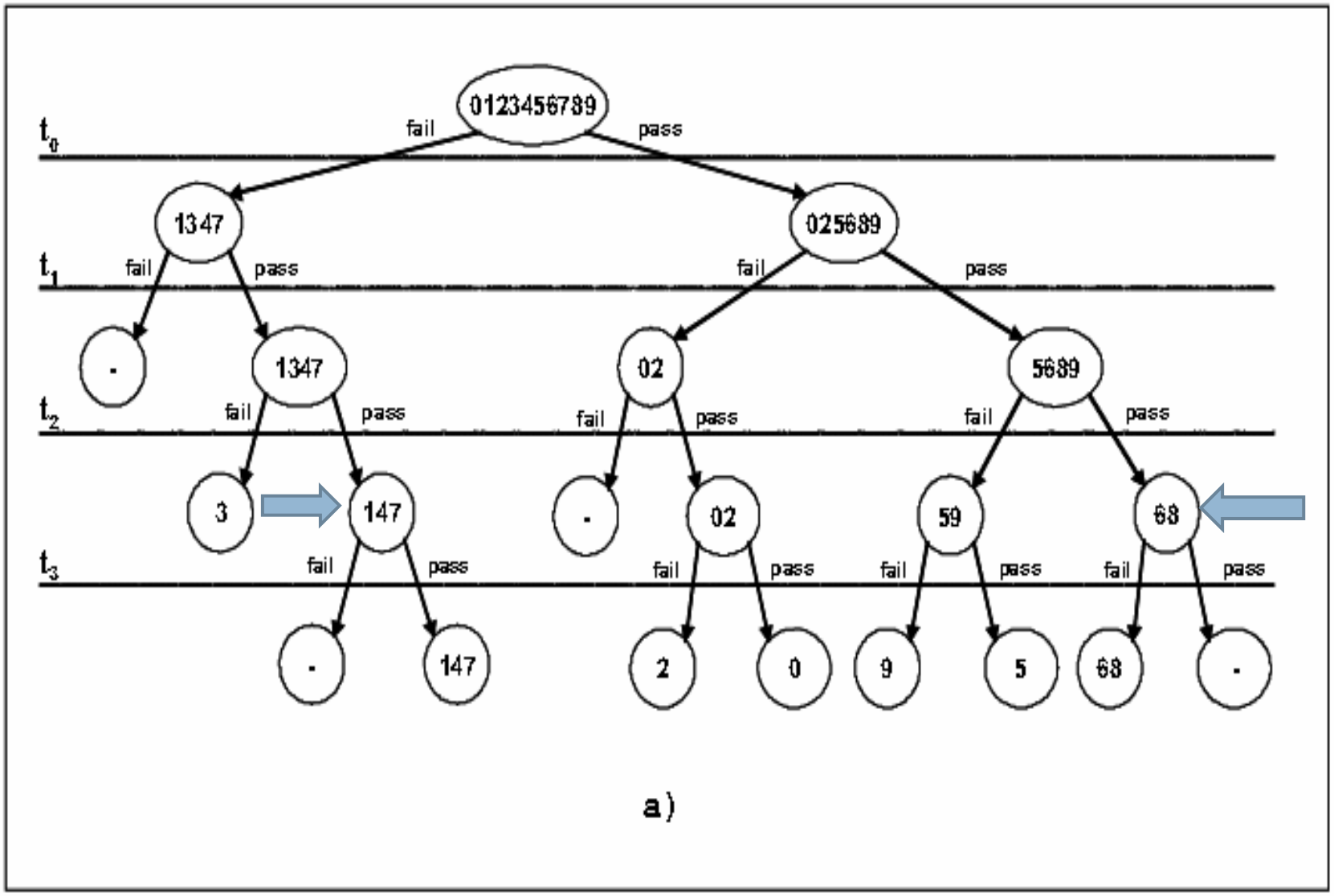
Coarse \ **Fine**	**0**	**1**	**2**
1	0:0,0 1;	1:0,0;	2;
6	0:3,0;	1;	

Dictionary Minimization

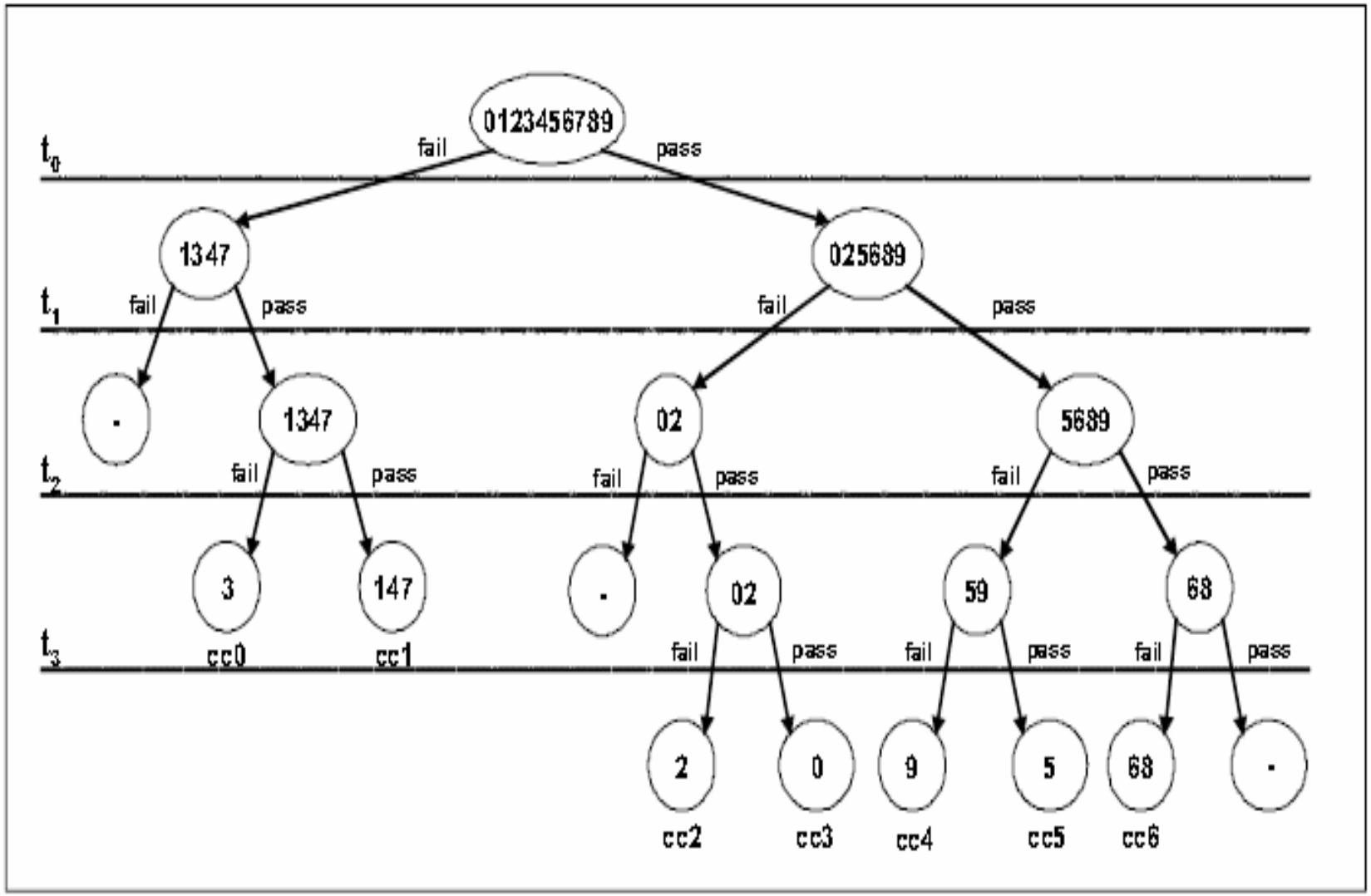
- The technique is based on the consideration that, in the pass/fail tree, faults are diagnosed by traversing the tree from its root to a leaf.
- The shorter these paths are, the smaller will be the pass/fail table in the fault dictionary.
- The proposed algorithm consists of:
  - A *dropping procedure,* performing a static length reduction of the tree's branches.
  - An *ordering procedure*, able to efficiently reorganize the tree structure to move leaves as close as possible to the tree root.

Dictionary Minimization (Dropping Procedure)

- The defined *dropping procedure* tries to reduce in length the tree branches owning leaves corresponding to equivalent classes that include more than one fault, by working directly on the pass/fail binary tree.
- Each branch leading to a multiple equivalent class is investigated and leaves reached by such paths are possibly moved up to previous tree levels.
- This tree modification is applicable when the leaf's father has 1 child only, thus not providing any additional diagnostic information
- It must not be performed if the tree edge to be eliminated is the only "fail" edge in the root-leaf path.

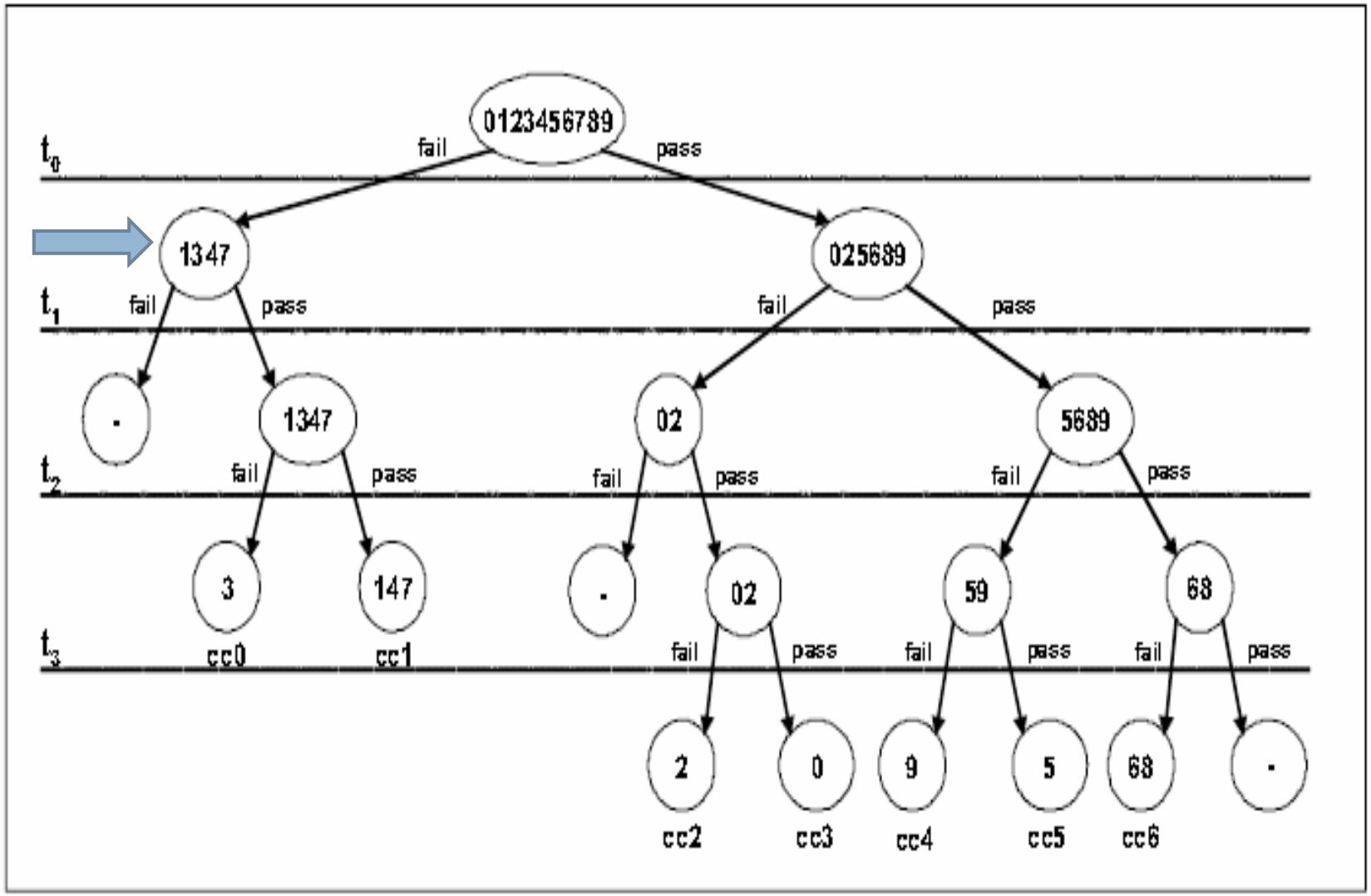

0123456789
fail
pass
t_0
1347
025689
t_1
fail
pass
fail
pass
-
1347
02
5689
t_2
fail
pass
fail
pass
fail
pass
3
147
-
02
59
68
t_3
fail
pass
fail
pass
fail
pass
fail
pass
-
147
2
0
9
5
68
-

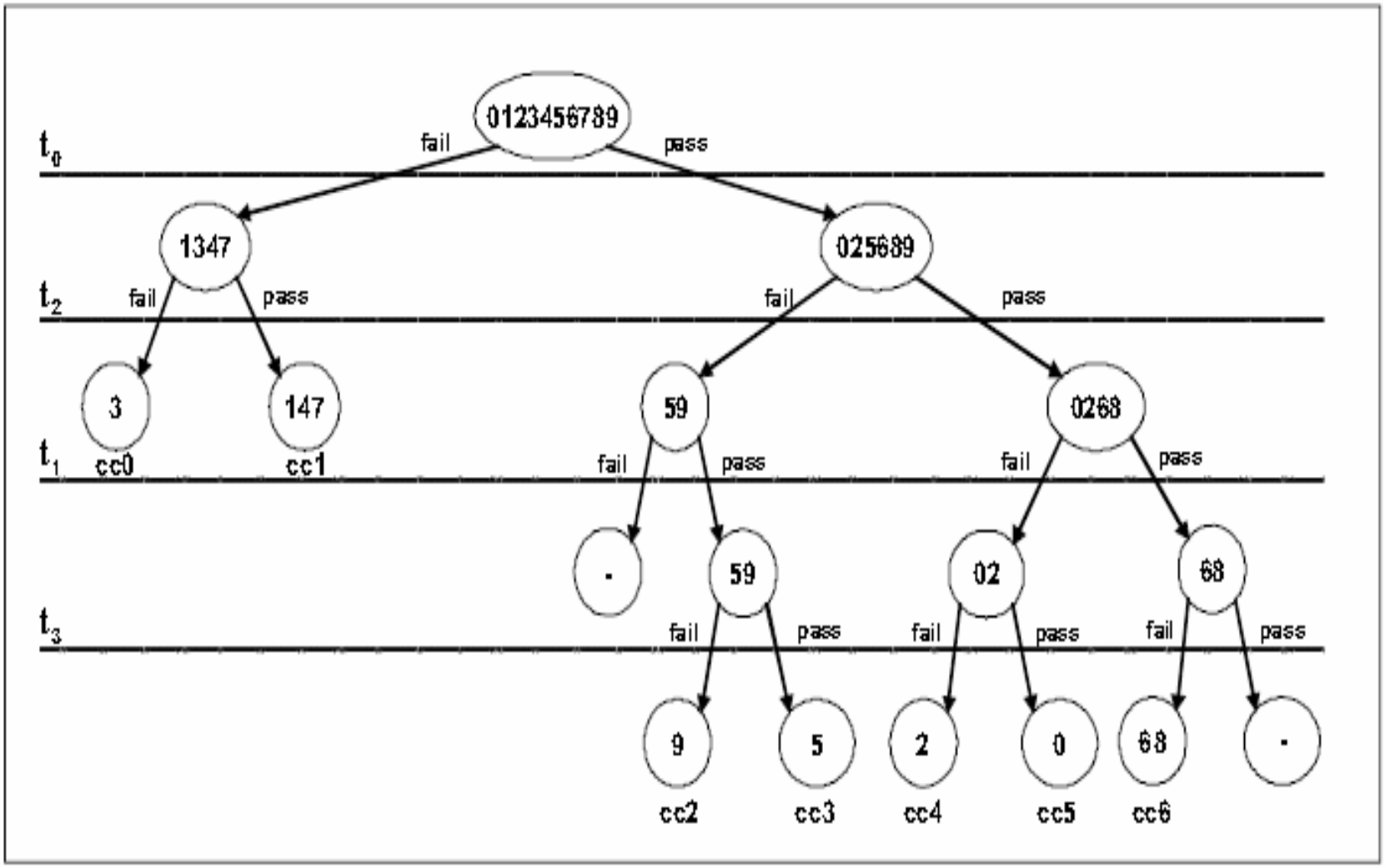
a)


0123456789
fail
pass
t_0
1347
025689
t_1
fail
pass
fail
pass
-
1347
02
5689
t_2
fail
pass
fail
pass
fail
pass
3
147
-
02
59
68
t_3
cc0
cc1
fail
pass
fail
pass
fail
pass
2
0
9
5
68
-
cc2
cc3
cc4
cc5
cc6

Dictionary Minimization (Ordering Procedure)

- The diagnostic expectation value offered by a test set is left unaltered when applying its patterns in a different order. This is called pattern inversion [7].
- From the tree size point of view, patterns inversion is advantageous if it is able to move forward in the test set a pattern useless for fault diagnosis.
- Nodes having only one child that generates two leaves are called *weak nodes* and their localization in the tree is one of the key points of the ordering procedure.

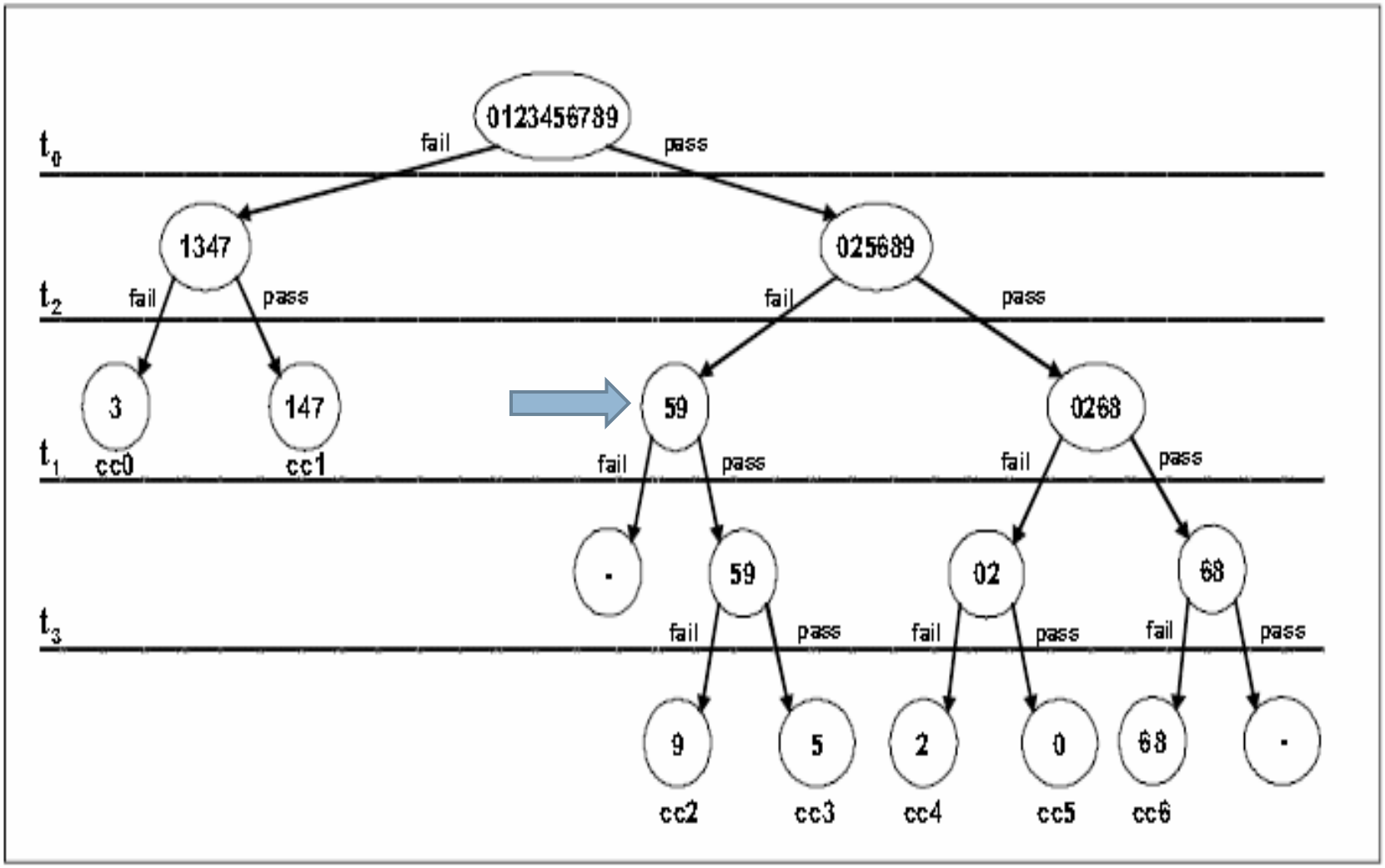

0123456789
fail
pass
t_0
1347
025689
t_1
fail
pass
fail
pass
-
1347
02
5689
t_2
fail
pass
fail
pass
fail
pass
3
147
-
02
59
68
t_3
cc0
cc1
fail
pass
fail
pass
fail
pass
2
0
9
5
68
-
cc2
cc3
cc4
cc5
cc6

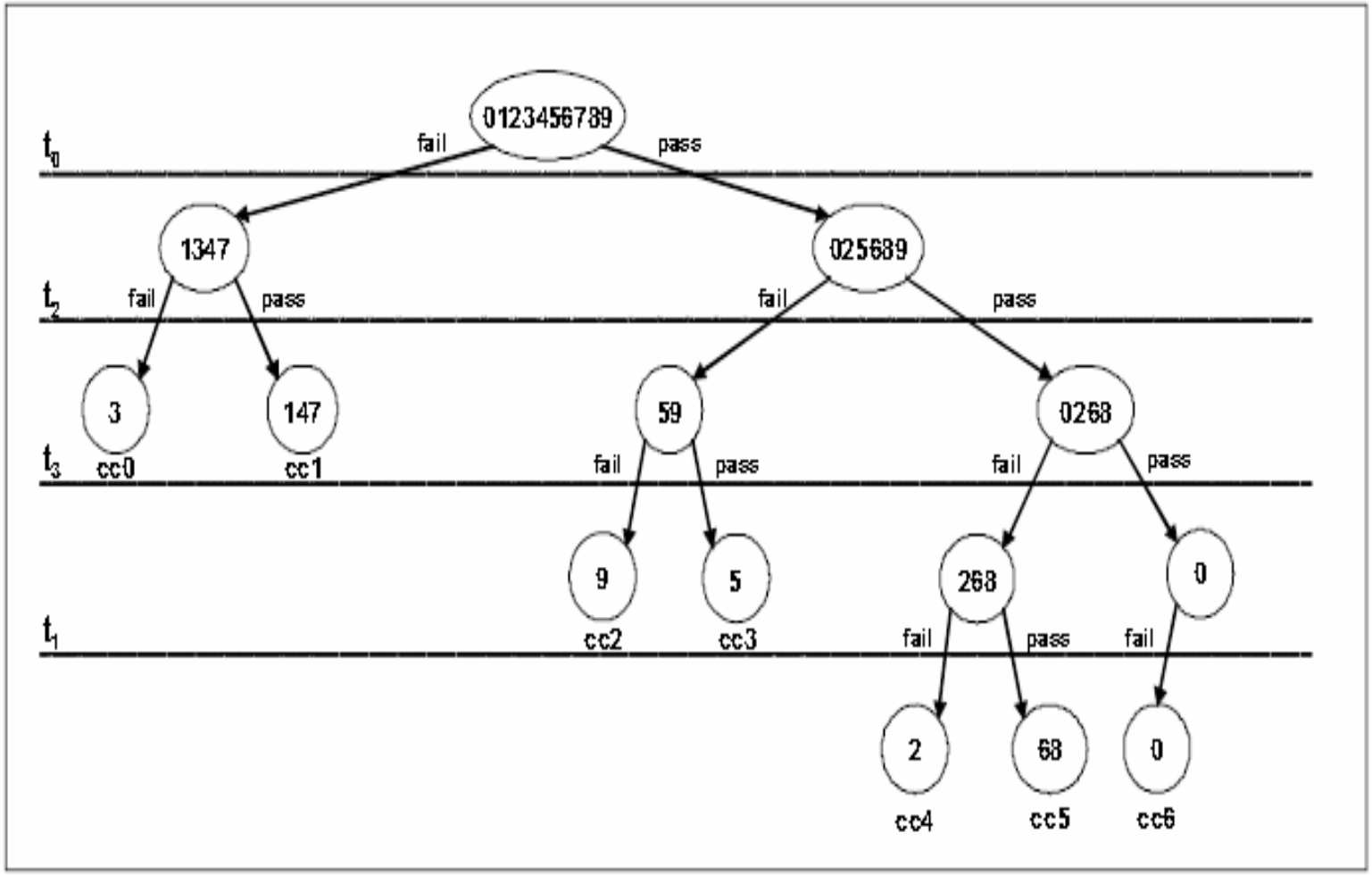

0123456789
fail
pass
t_0
1347
025689
t_2
fail
pass
fail
pass
3
147
59
0268
t_1
cc0
cc1
fail
pass
fail
pass
-
59
02
68
t_3
fail
pass
fail
pass
fail
pass
9
5
2
0
68
-
cc2
cc3
cc4
cc5
cc6

Dictionary Minimization (Ordering Procedure)

- Finding an optimal pattern sequence is a NP-hard problem, a greedy algorithm is proposed

```
n = min_weak_search(0);
while (n!=0)
        {
                invert(n);
                drop_tree(n);
                n = min_weak_search(n-1);
        }
```


0123456789
fail
pass
t_0
1347
025689
t_2
fail
pass
fail
pass
3
147
59
0268
t_1
cc0
cc1
fail
pass
fail
pass
-
59
02
68
t_3
fail
pass
fail
pass
fail
pass
9
5
2
0
68
-
cc2
cc3
cc4
cc5
cc6


0123456789
fail
pass
t_0
1347
025689
t_2
fail
pass
fail
pass
3
147
59
0268
cc0
cc1
t_3
fail
pass
fail
pass
9
5
268
0
cc2
cc3
t_1
fail
pass
fail
2
68
0
cc4
cc5
cc6

Dictionary Minimization (Ordering Procedure)

TABLE II: Pass/Fail table.

Coarse class	t_0	t_1	t_2	t_3
0	1	0	1	
1	1	0	0	0
2	0	1	0	1
3	0	1	0	0
4	0	0	1	1
5	0	0	1	0
6	0	0	0	1

TABLE IV: Pass/Fail table for the ordered patterns.

Coarse class	t_0	t_2	t_3	t_1
0	1	1		
1	1	0		
2	0	0	1	1
3	0	0	0	1
4	0	1	1	
5	0	1	0	
6	0	0	1	0

Experimental Results

TABLE VI: Fault dictionary tables size.

circuit	Coarse classes table [byte]	Fine classes table [byte]	Pass/Fail table [byte]		Δ size [%]
			Before patterns ordering	After patterns ordering	
s13207	57.4K	83.7K	41.2K	30.1K	26.94
s15850	67.6K	74.5K	52.8K	42K	20.45
s35932	207.2K	162.8K	247.2K	191.2K	22.66
s38584	198.9K	170.8K	229.9K	202.8K	11,78
s38417	197.7K	181.6K	191.9K	156.9K	17,93
b05	13.5K	57.6K	813	559.7	31.16
b06	1.1K	1.7K	288	260	9.72
b08	3.5K	3.2K	1.7K	1.3K	23.53
b09	3.4K	3.8K	1.6K	1.4K	12.50
b10	3.8K	3.5K	2.4K	1.9K	20.83
b11	13.3K	18.4K	18.1K	11.6K	35.91
b12	26.2K	24.4K	27.8K	21.4K	23.02
b13	6.5K	136K	431.7K	339.8K	21.29
b15	165.1K	136.1K	413.7K	339.8K	17.86
b17	555.6K	556.5K	2.4M	1.8M	25.00
b20	459.6K	537.0K	4.4M	2.6M	40.91
b21	470.5K	477.2K	5.4M	3.6M	33.34
b22	693.8K	683.7K	8.2M	5.2M	36.58

Conclusion

- In this a paper, a methodology for generating a reduced-size fault dictionary is presented.
- This procedure is based on the construction of an effective dictionary representation as a pass/fail diagnostic tree enhanced with few additional information about faulty outputs.
- The pass/fail tree is manipulated in order to find a new patterns order minimizing the root-leaf path length, thus reducing the number of stored information and the dictionary access time.
- The effectiveness of the proposed strategy has been experimentally proved on a set of benchmarks circuits included in the iscas-89 and itc-99 benchmark sets.

References

- [1] I. Pomeranz, S.M. Reddy, "On dictionary-based fault location in digital logic circuits", IEEE Transactions on Computers, Volume 46, Issue 1, Jan. 1997 Page(s):48 – 59
- [2] V. Boppana, I. Hartanto, W.K. Fuchs, "Full fault dictionary storage based on labeled tree encoding", VLSI Test Symposium, 1996. Page(s):174 – 179
- [3] B. Chess, T. Larrabee, "Creating small fault dictionaries", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume 18, Issue 3, March 1999 Page(s):346 - 356
- [4] S. Chakravarty, V. Gopal, "Techniques to encode and compress fault dictionaries" VLSI Test Symposium, 1999. Page(s):195 – 200
- [5] J.A. Waicukauski, E. Lindbloom, "Failure diagnosis of structured VLSI", IEEE Design & Test of Computers, Volume 6, Issue 4, Aug. 1989 Page(s):49 - 60
- [6] I. Pomeranz, "On pass/fail dictionaries for scan circuits", Asian Test Symposium, 2001. Page(s):51 – 56
- [7] H. Chang, E. Manning, G. Metze, "Fault diagnosis of digital systems", Wiley Interscience, New York, 1970