ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2009, Vol. 38, No. 4

TEST QUALITY ASSESSMENT BASED ON SMALL DELAY DEFECTS

Eduardas Bareiša, Vacius Jusas, Kęstutis Motiejūnas, Rimantas Šeinauskas, Žydrūnas Tamoševičius

Software Engineering Department, Kaunas University of Technology
Studentų 50-404, LT-51368 Kaunas, Lithuania
e-mail: vacius.jusas@ktu.lt

Abstract. The quality of delay testing focused on small delay defects is not known when transition fault model is used. The paper presents a method that evaluates the quality of the delay test according to the covered paths of the circuit and constructs the paths, which could be used as the input to the path delay test generator. All the constructed paths are testable. The complexity of the circuit has no direct impact on the path construction. The path construction is based on the information provided by TetraMAX transition fault simulator. The transition fault simulator forms a text file that contains the complete information on the propagation of the transitions along the lines of the circuit. The experimental results demonstrate the ability to assess the quality of the delay test according to the covered paths.

Keywords: path delay test, transition fault test, transition fault simulator.

1. Introduction

Rapidly shrinking feature sizes raise the spectrum of new types of defects, and increasing gate counts have increased the number of locations where such defects can occur. The presence of some random defects does not affect a circuit's operation at slow speed while it may cause circuit malfunction at rated speed. This kind of defect is called the delay defect. The purpose of a delay test is to verify that the circuit operates correctly at a desired clock speed. Although application of stuck-at fault tests can detect some delay defects, it is no longer sufficient to test the circuit for the stuck-at faults alone. The pair of test patterns is used to detect delay faults. The first pattern sets the initial values on the inputs of the circuit; the second pattern launches the transition.

Mainly, two types of delay fault models are used: the transition fault model [3], and the path delay fault model [12]. The transition fault model assumes that the delay fault affects only one gate in the circuit, and the extra delay caused by the fault is large enough to prevent the transition from reaching any primary output within the specification time. In other words, the transition fault can be detected on any sensitized path through the fault site. The transition fault coverage is measured as the percentage of faults which are detected by a test set. The main advantage of transition fault model is that the number of faults in the circuit is linear to the number of gates. Also, the stuck-at fault test generation procedure can be easily modified for transition fault test generation [13].

Under the path delay fault model a circuit is considered faulty if the delay of any of its paths exceeds the specification time. The path delay fault model is more realistic in modeling physical delay defects because the model can also detect small distributed delay defects caused by process variation, or the combination of local and distributed delay [7]. However, a major limitation of this fault model is that the number of paths in the circuit (and therefore the number of path delay faults) can be exponential to the number of gates. For example, ISCAS85 benchmark circuit c6288, a 16-bit multiplier, has close to 10^{20} paths [8].

Many techniques have been used to reduce the number of paths that must be tested in the path delay fault model. The simplest idea is to test the only longest paths, which have the maximum delays. These paths are also called critical paths because they dominate the performance of the circuit [6]. However, circuit optimization tends to compress the distribution of path delays in a circuit, so many paths are close to the maximum delay [14]. Because of manufacturing process variation, any of these paths can be the actual longest path. Therefore, a group of longest paths must be selected for testing.

The transition fault model and the path delay fault model present two orthogonal approaches. It was then hoped that the combination of these two models can capture most of the delay defects and ensure the circuit performance. Many efforts were made to this direction [4], [8], [9], [10], [11], [15].

In order to have a fault model that can provide a complete topological coverage and, simultaneously, exercise the worst-case timing scenarios, researchers have proposed methods to select critical paths that cover all transition fault sites [8], [10], [11]. A common issue among these methods is that many of the selected critical paths may not be testable. A path is said to be testable if a rising/falling transition can propagate from the primary input to the primary output associated with the path under certain sensitization criteria [8]. If a path is not testable, it is called an untestable path.

The possible solution to the problem of selecting only testable paths is to pre-select a number of long paths passing through each site and then, following the order of their timing lengths, to check one by one until the first testable path can be identified [10]. This path enumeration approach can be inefficient, depending on the number of paths that have been checked before a testable path is identified. New techniques [8], [11] were developed to identify the testable critical paths without using the path enumeration approach. Many techniques have been used to significantly reduce the search space [8].

Selecting critical paths to cover all sites is different from detecting each transition fault through the longest testable (propagation) path. In the former case, all paths are ranked together based on their timing lengths and hence, there is only one ordering among all paths. In the latter case, all paths from a transition fault site are ranked together. Each site has its own ranking of the paths. An ATPG called POTENT [9] was first proposed for the path-oriented transition fault model. This ATPG was based on a greedy path expansion heuristic. A rather simplified metric was also proposed to evaluate the quality of the resulting tests. With their fixed delay model, the worst-case slacks were characterized by simulation as the measurement for quality.

The approach similar to [9] was developed in [15], where Yang et al presented a test generation tool targeting on a path-oriented transition fault model. Under this model, a transition fault is detected through the longest testable path. False path pruning technique is used to identify the longest testable path through each fault site. The quality of test patterns is evaluated by statistical delay simulation on the randomly generated samples of delay defects. Such a metric does not show the real quality of the test patterns.

Gupta and Hsiao [4] moved further in joining up the path delay test patterns and transition fault test patterns. The employed approach is clearly expressed in the objective of the paper [4] – in order to have a high quality delay test, it needs to have high robust path coverage, high non-robust path coverage and high transition fault coverage. The delay test patterns are generated in the enumerated order. Every next step takes into account the results of the previous step. The implication-based technique is used for the removal of all the untestable robust paths. But the authors confessed – since the implication engine is not complete they cannot conclude the detectability about the paths that were not detected as untestable.

The exceptional attention is shown to the critical path testing in all the reviewed papers, because critical path selection is an indispensable step for delay test and timing validation [6]. Only a few of the papers [4], [9], [15] pay some attention to the transition delay testing as supplementary to the critical path testing. Liou et al. [6] proved that the problem of critical path selection is computationally intractable. The practical heuristics should be used for the critical paths selection. Therefore, this way will always meet difficulties and will not be reliable. Qiu et al [7] suggested the different strategy according to which the transition fault tests should be applied in the first place. Only then the path delay tests should be applied to the longest paths. This strategy can be used as the basis for the delay test pattern generation. The transition delay fault model is well known in test delay quality evaluation. The transition delay test is targeted to detect large delay defects. But the transitions propagate along the paths of the circuit, and such a test can detect some small delay defects [2]. Therefore, the quality of the transition test should be evaluated according to the small delay defects, as well.

The objective of the paper is to present the approach of the test delay quality assessment, which is based on small delay defects. The rest of the paper is organized as follows. In Section 2, we present the paths identification approach, which enables to identify the paths that the transition fault test exercises. We report the results of the experiment in Section 3. We finish with conclusions in Section 4.

2. Path identification approach

The transition delay fault model recognizes delay propagation when a transition fault in a two-time-frame is identified as a logic value flipping at a primary output. As it can be confirmed by logic simulation, we usually don't have to know the propagation paths. But the transition fault simulator typically contains the information on the transition faults covered by the test patterns pair and propagation of them along the paths of the circuit. On the base of this information, the propagation paths can be identified. The identification of the paths would allow assessment of the quality not only for the large delay faults, but for the small delay faults as well.

One of the most known commercial simulators is TetraMAX transition fault simulator. This simulator can provide the detailed information on the propagation of the transition faults along the lines of the circuit. The appropriate commands have to be included into the script file that drives the fault simulator. The transition fault simulator forms the text file that contains the complete information for every pair of test patterns. The file includes the information on the active transitions only. The active transition is such a

one that enables detection of the appropriate transition fault. The format of the file is very suitable for the lexical analyzer.

Consider an example of the circuit presented in Figure 1. The description of the circuit in Verilog hardware language is presented in Figure 2. The list of the structural paths of this circuit is shown in Table 1. The number that is shown in the first column is used as the label of the appropriate path. A path is represented by the list of the ports of the gates through which it passes. The ports of the gates are enumerated in the order that the path connects them. Every path starts on the primary input and only then it connects to the input port of the gate. Next, the path passes to the output port of the gate. Then it connects to the input port of the following gate. Every path ends on the primary output. The length of the path shown in the third column of Table 1 is the doubled number of the gates, through which it passes, plus primary input and primary output. Therefore, the length of the path always is even. The number of the gates is doubled, because the input port and the output port of the gate are counted separately.

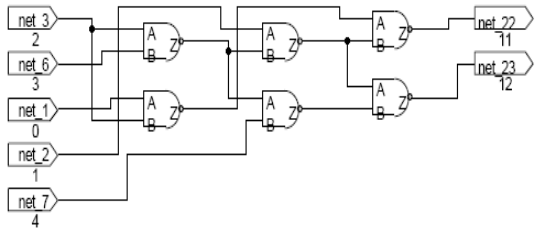

Figure 1. Circuit c17 from the benchmark suite ISCAS85

```
module c17 (net_1, net_2, net_3, net_6,
            net_7, net_22, net_23);
input      net_1, net_2, net_3,
           net_6,net_7;
output     net_22,net_23;
wire       net_10, net_11, net_16,
           net_19, net_22, net_23;

NAND_2  d1  (net_10, net_1,  net_3 );
NAND_2  d2  (net_11, net_3,  net_6 );
NAND_2  d3  (net_16, net_2,  net_11);
NAND_2  d4  (net_19, net_11, net_7 );
NAND_2  d5  (net_22, net_10, net_16);
NAND_2  d6  (net_23, net_16, net_19);
endmodule
```

Figure 2. Verilog description of circuit c17

Consider the path No 1 that is shown in Table 1. The path starts on the primary input *net_2*. Next, it connects to the input port of the gate *d3/A*. The notation "*d3/A*" has to be read in the following way: *d3* denotes the name of the gate, *A* denotes the name of the input port of the gate. If we look at the Verilog netlist in Figure 2, we would not see the explicit names of the ports of the gates, because the positional syntax is used. Following the name of the gate (*d3*), inside parentheses the output and input ports of the gate are written. The output of the gate is always on the left inside the parentheses. In the positional syntax, the names of the ports are extracted from the library.

Consider the line "*NAND_2 d3 (net_16, net_2, net_11)*" in Figure 2. The line *net_16* is connected to the output port, which is named *Z*. The line *net_2* is connected to the first input port that is named *A*, and the line *net_11* is connected to the second input port that is named *B*. The path is identified on the base of these notations.

Table 1. Structural paths of the circuit c17

No	Structural path	Length
1	net_2, d3/A, d3/Z, d5/B, d5/Z, net_22	6
2	net_2, d3/A, d3/Z, d6/A, d6/Z, net_23	6
3	net_3, d2/A, d2/Z, d3/B, d3/Z, d5/B, d5/Z, net_22	8
4	net_3, d2/A, d2/Z, d3/B, d3/Z, d6/A, d6/Z, net_23	8
5	net_3, d2/A, d2/Z, d4/A, d4/Z, d6/B, d6/Z, net_23	8
6	net_3, d1/B, d1/Z, d5/A, d5/Z, net_22	6
7	net_6, d2/B, d2/Z, d3/B, d3/Z, d5/B, d5/Z, net_22	8
8	net_6, d2/B, d2/Z, d3/B, d3/Z, d6/A, d6/Z, net_23	8
9	net_6, d2/B, d2/Z, d4/A, d4/Z, d6/B, d6/Z, net_23	8
10	net_1, d1/A, d1/Z, d5/A, d5/Z, net_22	6
11	net_7, d4/B, d4/Z, d6/B, d6/Z, net_23	6

Table 2. Transition test patterns

No of pair	net_1	net_2	net_3	net_6	net_7	No of paths
1	0	0	0	0	0	6r, 10r
	1	0	1	1	0	
2	0	0	0	0	0	1r, 2r
	0	1	0	0	1	
3	0	1	0	0	0	
	0	0	1	1	0	
4	1	0	1	1	0	11r
	0	0	0	0	1	
5	0	0	0	0	1	3r,4r,5r, 7r,
	1	1	1	1	0	8r, 9r, 10r
6	0	0	1	0	0	11r
	1	0	0	0	1	
7	1	1	0	0	0	1f, 2f, 10f
	0	0	1	0	0	
8	1	0	0	1	0	
	0	1	1	1	1	
9	0	1	1	1	1	3f, 4f
	0	1	0	1	0	
10	0	1	1	1	1	7f, 8f
	0	1	1	0	1	
11	1	1	0	0	1	1f, 2f, 11f
	0	0	0	1	0	

The transition test patterns generated to detect transition faults of the circuit *c17* (Figure 1) and the

paths covered by these test patterns are shown in Table 2. The transition fault coverage is 100%. The table contains 11 pairs of test patterns, which are enumerated according to the inputs *net_1*, *net_2*, *net_3*, *net_6*, and *net_7*. The pairs of test patterns are numerated in the column under name "No of pair". The numbers of paths shown in the columns under name "No of paths" indicate the paths enumerated in Table 1. The paths can be either rising or falling. The letter next to the path number indicates the direction of the path (r – rising, f – falling).

TetraMAX transition fault simulator, when the appropriate commands are included into the script file, forms the listing, which includes the static information on the transition propagation along the paths of the circuit. The output generated by TetraMAX transition fault simulator for the first pair of test patterns is presented in Figure 3. Every pair of test patterns has its own portion of the lines that are similar to those displayed in Figure 3.

```
str   DS   net_1
str   --   d1/A
stf   DS   d1/Z
stf   --   d5/A
str   DS   d1/B
str   DS   net_22
str   --   d5/Z
str   DS   net_3
```

Figure 3. Transitions of the first pair of the test patterns

Notation "*str*" denotes rising transition, notation "*stf*" denotes falling transition. The paths can be identified using the information of the transition propagation and the netlist of the circuit presented in Figure 2. The form of the netlist is quite good for the manual identification of the paths as it was accomplished in Table 2, but this form is complicated for the lexical analyzer. Luckily, the TetraMAX program can prepare the netlist in the format (Figure 4) similar to the format presented in Figure 3. The data organized according to this format can be easily distinguished as the input data by the lexical analyzer.

```
d1 (5)  NAND (NAND_2)
     A          I  0-net_1
     B          I  2-net_3
     Z          O  8-/d5/A
```

Figure 4. Port list of gate d1

Figure 4 presents an excerpt of the listing dedicated to gate *d1*. In this listing, a separate line is allocated for every port of the gate. Every structural unit of the circuit is presented by the separate line as well. The format of the ports of the circuit is the same in both listings. Therefore, the information on the transition propagation in one listing and the extended netlist of the gates in the other listing can be easily related by the lexical analyzer.

Let's trace the information for the first pair of test patterns in the listing shown in Figure 3. The rising transition "*str*" starts at the *net_1*. Then, we check the netlist in the listing of Figure 4 and we find that the *net_1* is the primary input and it is connected to the gate *d1/A*. The rising transition "*str*" is present at *d1/A*. Next, the falling transition is present at *d1/Z*, which is the output of gate *d1*. Then, we look for the netlist and we find that *d1/Z* is connected to *d5/A*. The transitions are present at *d5/A*, *d5/Z* and *net_22*, which is the primary output, and it is connected to *d5/Z*. So, the first path, which has the number 10 in Table 1, is identified. In a similar way, the second path, which starts at *net_3*, has the number 6 in Table 1 and follows the same lines after gate *d1* is identified. Both paths are rising. The rising transition is present at the primary input *net_6*, but this transition is not included into the listing because no transition faults can propagate along the path starting at the primary input *net_6*.

```
str   DS   d5/B
stf   DS   net_22
stf   --   d5/Z
str   DS   d3/Z
str   DS   d6/A
stf   DS   net_23
stf   --   d6/Z
```

Figure 5. Transitions of the third pair of the test patterns

The information presented on the transition propagation is not always such comprehensive as it was for the first pair of the test patterns. For example, consider the third pair of test patterns. This pair of test patterns starts the transitions (Table 2) at the primary inputs *net_2*, *net_3*, *net_6* and detects some transition faults. But the listing includes no transition at the primary inputs (Figure 5). The transitions for faults detection start at the faults site. Therefore, the transitions at the primary inputs were not included into the listing. Consequently, no path can be identified. In such a way, we loose some information on the paths.

```
stf   DS   net_1
stf   --   d1/A
stf   DS   d3/B
str   DS   net_3
str   DS   d2/A
stf   DS   d2/Z
stf   DS   d4/A
```

Figure 6. Transitions of the eighth pair of the test patterns

```
stf   DS   d1/B
stf   DS   net_3
str   DS   net_23
str   --   d6/Z
str   DS   net_7
str   --   d4/B
stf   DS   d4/Z
stf   --   d6/B
```

Figure 7. Transitions of the sixth pair of the test patterns

The different situation is with the eighth pair of test patterns (Figure 6). No transition is present at the primary outputs because the transitions faults are detected non-robustly. But the result is the same – no path can be identified. The similar situation is with the sixth pair of test patterns (Figure 7). The falling transition starts at the primary input *net_3*, which is connected to the *d1/B*. But no transition propagates to

the output of gate *d1*. The transition fault that could be present at the *net_3* is detected non-robustly along the path. Again, the path can not be identified. But the rising transition, which starts at the primary input *net_7,* allows the identification of the path because it ends as the rising transition at the primary output *net_23*.

We could notice that all the structural paths of the circuit are covered already after analysis of the fifth pair of test patterns. If we are interested only in the structural paths, that information would be enough. But the obtained information on the transitions allows accomplishing the deeper analysis. It makes possible differentiating between the rising and falling paths. All the paths till the seventh pair of test patterns are rising. The remaining pairs of test patterns contribute the falling paths only. The falling transitions do not start on the paths No. 5, 6, and 9. The rising paths No 10 and No 11 are covered two times. The falling paths No 1 and No 2 are covered two times, as well. So, the total number of the covered paths is 23, meanwhile 19 paths of them are unique, because they are covered only once.

As it was shown in the Introduction, one of the most important qualities of the path delay test is the number of the longest testable paths passing through each fault site. But the transition test that is under consideration for the detection of small delay faults does not target explicitly the longest paths of the circuit. Therefore, the found longest path under transition test can be not the actual longest path passing through the fault site. Thus, we introduce the notion of the test longest path, which will imply the longest path under transition test.

Let's count the number of the test longest paths passing through each fault site for our example. Circuit *c17* contains 6 gates; each of them has 3 ports. So, the total number of fault sites is 18. No new path can cover the output ports of the gates. Consequently, the number of fault sites is decreased to 12. No new test longest paths cover the ports *d5/A*, *d5/B*, *d6/A*. The ports *d1/B*, *d4/A*, and *d6/B* are covered only once by the test longest paths. All the remaining ports are covered twice by the test longest paths. So, the total number of the test longest paths passing trough each fault site is 15.

Now, we can define an algorithm for the identification of the sensitized paths in the transition delay test. The algorithm includes the following steps:

1. Prepare the transition fault simulation script file and include the commands for the formation of detailed listing that would contain information on the transition propagation along the lines of the circuit.
2. Simulate transition faults.
3. Start the lexical analyzer that takes as the input two text files (information on the transition propagation and netlist of the circuit) from the fault simulator.
4. Obtain the statistics on the sensitized paths by the transition fault test from the lexical analyzer.

Since the test longest path is not always the actual longest path of the circuit, the number of the longest paths cannot be the correct assessment of the test quality for covering small delay faults. For example, if one test exercises the one long path, another test exercises two shorter paths. The first test has to be better for covering small delay faults, because the two shorter paths may never fail. Therefore, the main indicator of the coverage of the small delay faults has to be the average of the length of the test longest paths. But this value could not be used on itself; it has meaning only in comparison of two delay tests.

3. Experimental results

The path identification program based on the approach presented in the previous section was implemented. The following text files are used as an input to the program: detailed netlist of the circuit and detailed information on the propagation of the transitions along the lines of the circuit. The output of the program is the statistics on the paths covered by the transition fault test and the file that contains identified paths in the format acceptable by TetraMAX path delay test generator.

Table 3. TetraMAX transition test patterns

Circuit	Test patterns	Fault coverage (%)	Test longest paths	Average of lengths	Coverage by paths (%)
C432	140	100	193	15,55	69,75
C499	221	100	221	22,53	52,99
C880	152	100	398	18,18	77,30
C1355	308	100	246	24,31	56,48
C1908	314	100	577	27,53	70,07
C2670	260	100	780	19,95	74,70
C3540	408	100	653	30,12	46,95
C5315	281	100	2138	15,61	64,25
C6288	112	100	980	47,77	62,67
C7552	461	100	2096	25,88	66,24

The experiments were carried out on the ISCAS85 benchmark circuits. Four different transition test pattern sets were used for the experiments. The results are presented in Table 3, Table 4, Table 5, and Table 6. Each table has the same headings for 6 columns: circuit name, the number of transition test patterns, transition fault coverage, the test longest paths passing through each fault site, the average of the length of the test longest paths, transition fault coverage by the test longest paths. The information presented in the fourth and the sixth columns requires some further explanation. The test longest paths passing through each fault site were identified according to the presented approach. The total number of test longest paths is

presented in the fourth column. These paths cover not all the transition faults. The last column in all the tables shows what percentage of the transition faults is covered by the identified test longest paths.

Table 3 displays the results of paths identification according to transition test patterns generated by TetraMAX. All the other tables show the results of delay test patterns generated at the functional level. Table 4 reports the results of path identification according to the functional delay single-input transition (SIT) test patterns constructed from functional stuck-at test. The rule of the functional delay test construction was the following. Suppose we have an input pattern that detects q pin pair faults [1] (Mode 1). Thus, for detection of the corresponding q functional delay faults, at most w pairs of input patterns are built of this pattern (signal transition on one input can cause signal transitions on s outputs, consequently, only one pair of input patterns is needed for detection of s functional delay faults). The constructed pairs of test patterns possess the change of signal value only on one input. Therefore, they are SIT tests and functional robust.

Table 4. Robust SIT delay functional test constructed from functional stuck-at test

Circuit	Test patterns	Fault coverage (%)	Test longest paths	Average of lengths	Coverage by paths (%)
C432	348	95,56	260	21,31	88,52
C499	5180	94,40	674	33,48	93,32
C880	1001	98,91	644	23,48	98,58
C1355	5162	97,13	670	35,42	95,01
C1908	2359	95,24	1019	42,37	93,37
C2670	1820	96,51	1070	25,18	93,91
C3540	1457	83,08	1205	42,17	69,89
C5315	4950	98,41	3320	24,39	88,70
C6288	1065	99,75	2520	90,36	96,50
C7552	5801	99,21	3232	29,91	83,51

Table 5 shows the result of paths identification according to the functional delay multi-input transition (MIT) test patterns constructed from functional stuck-at test. The rule of the functional delay test construction is the following. Every input pattern, which detects pin pair faults, is transformed only into one input pattern pair in such a way: the signal value transition takes place on every input that is associated with pin pair fault detection on the considered test pattern [1] (Mode 3). Table 6 demonstrates the results of path identification according to the functional delay MIT test patterns generated by the method presented in [5]. In this approach, the delay faults are detected in the function-robust and function-non-robust manner. Such a versatility of the approach allows obtaining high transition fault coverage and quite a small number of transition test patterns.

Table 5. Robust and non-robust MIT delay functional test constructed from functional stuck-at test

Circuit	Test patterns	Fault coverage (%)	Test longest paths	Average of lengths	Coverage by paths (%)
C432	117	84,28	179	13,02	62,86
C499	1077	91,44	420	27,07	73,94
C880	381	90,9	369	18,96	65,81
C1355	1011	92,34	406	29,09	76,91
C1908	620	81,2	241	21,89	28,77
C2670	448	90,44	519	22,93	49,85
C3540	515	84,42	523	35,73	34,31
C5315	1169	97,35	1763	17,68	52,55
C6288	268	98,39	576	39,10	34,73
C7552	2115	97,84	2300	27,64	67,75

Table 6. Generated MIT functional delay test

Circuit	Test patterns	Fault coverage (%)	Test longest paths	Average of lengths	Coverage by paths (%)
C432	230	96,08	226	14,61	77,19
C499	1774	98,55	422	27,12	74,78
C880	601	99,67	525	20,28	88,35
C1355	1698	96,73	412	29,05	77,74
C1908	1128	95,22	693	35,46	81,65
C2670	831	99,2	933	24,01	87,15
C3540	850	92,05	816	36,21	49,97
C5315	2098	99,56	3006	19,36	80,54
C6288	427	99,63	1142	43,92	61,59
C7552	3167	99,17	2998	29,85	82,79

We used three estimates to assess the quality of the transition fault test according to the small delay defects. These estimates are the following: the number of test longest paths passing through each fault site, the average length of the test longest paths, the transition fault coverage by the test longest paths. We suppose that the most important one is the transition fault coverage by the test longest paths. This indicator allows the quality assessment of the delay test according to the small delay defects without comparison with other delay tests. The next two indicators – the average length of the test longest paths and the number of the test longest paths passing through each fault site – have the meaning in comparison with other delay tests only. When two tests are compared, the better test is such one, the transition fault coverage of which is higher. If fault coverage of two tests is almost the same, the values of two other indicators have to be taken into account. Then the better test is such one, the average length of the test longest paths of which is

larger. The total number of the test longest paths can be larger but that does not mean the better test, because one long path is better than two short paths. The indicator of the average length of the test longest paths allows revealing this false impression.

Now we can compare the tests used in our experiments according to the considered indicators. The easiest way to compare the results obtained from different test generation modes is to draw a chart. The comparison of the transition fault coverage by the test longest paths is presented in Figure 8.

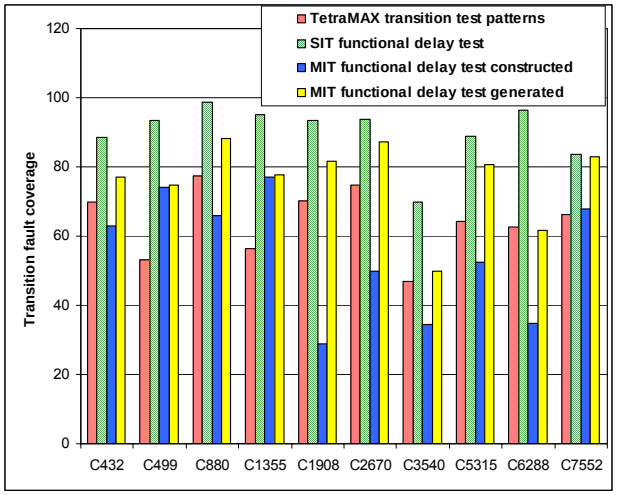

Figure 8. Comparison of the transition fault coverage by the test longest paths

As we can see from Figure 8, the SIT functional delay test is better for every circuit. Especially it outperforms the other test generation modes for the circuit C6288, which has the largest number of paths. The superiority of the SIT test could be explained by the reason that the SIT test covers the robust paths only. The implemented approach can identify the robust paths only, as well. MIT functional delay test exercises both the robust and non-robust paths. The TetraMAX transition delay test exercises the robust and non-robust paths, as well. Additionally, the TetraMAX transition delay test covers the paths that start at the fault site. Such paths are not counted, too. Therefore, the TetraMAX transition delay test is not as good as the generated MIT functional delay test.

The value of the test is estimated not only by its quality, when the higher number means the better quality. The important criterion of the test value is the number of the test patterns. In this case, the smaller number means the better value of the test. The comparison of the number of the generated test patterns is presented in Figure 9. We can see that the number of the SIT test patterns is incomparable large. Consequently, despite the best indicator of the transition fault coverage by the test longest paths, the SIT test could not be recognized as the best choice for covering path delay faults. Keeping in mind the transition fault coverage by the test longest paths and the number of the generated test patterns, the MIT generated test, which is represented by the fourth column for every circuit in charts, is the best choice. As we can see from Figure 8, the MIT generated test is not as good as the SIT test, but the MIT generated test outperforms the TetraMAX test for every circuit, except the circuit C6288. As we can see from Figure 9, the MIT is much better than the SIT test.

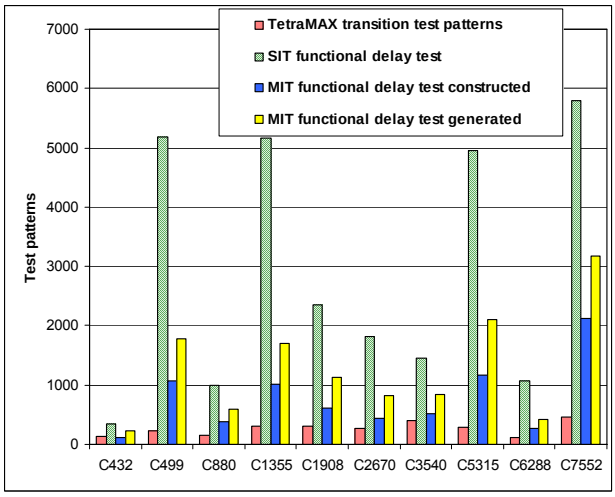

Figure 9. Comparison of the number of the test patterns

4. Conclusion

The newly introduced criterion, the transition fault coverage by the test longest paths, allows the quality assessment of the transition delay test according to the requirements of path delay test. But this criterion should not be used only. The number of the generated test patterns has to be taken into account. Balancing these two criteria, where the first one is more important than the second one, we can conclude that the best choice for covering path delay faults among considered approaches is the MIT generated test.

The output of the implementation is not only the quality assessment but it constructs the paths in the format, which is acceptable as the input to the TetraMAX program that generates the path delay test. That strengthens the value of the presented approach, because it could replace the tool of the path construction like PrimeTime of Synopsys system. All the paths constructed using our approach are testable.

The obtained results underline the value of test generation at the functional level for the path delay test. The generation at the functional level is oriented to the covering of all the possible paths between the primary inputs and the primary outputs. The transitions, which propagate along the paths of the circuit, start at the primary inputs. Meanwhile, the transition test generation cares about that the transition would start at the fault site only. This is the main reason why the transition test generated by TetraMAX program covers fewer paths.

References

[1] **E. Bareiša, V. Jusas, K. Motiejūnas, R. Šeinauskas.** Functional Delay Test Construction Approaches. *Elektronika ir elektrotechnika = Electronics and electrical engineering*. 2007, *No*. 2(74), 49 - 54.

[2] **E. Bareiša, V. Jusas, K. Motiejūnas, R. Šeinauskas.** Properties of Variable n-Detection Functional Delay Fault Test. *Information Technology and Control*, 2008, *Vol*. 37, *No*. 2, 95 - 100.

[3] **Z. Barzilai, B.K. Rosen**. Comparison of AC Self-Testing Procedures. *Proceedings of the IEEE International Test Conference*, Philadelphia, PA, Oct. 1983, 89-94.

[4] **P. Gupta, M.S. Hsiao.** High Quality ATPG for Delay Defects. *Proceedings of the IEEE International Test Conference, September 2003*, 584-591.

[5] **V. Jusas, K. Motiejūnas.**. Generation of Functional Delay Test with Multiple Input Transitions. *Information Technology and Control*, 2007, *Vol*. 36, *No*. 3, 259 - 267.

[6] **J.J. Liou, L.C. Wang, K.T. Cheng.** On Theoretical and Practical Considerations of Path Selection for Delay Fault Testing. *Proceedings of the IEEE/ACM International Conference on Computer Aided Design*, *San Jose, CA, Nov*. 2002, 94-100.

[7] **W. Qiu, X. Lu, J. Wang, Z. Li, D.M.H. Walker, W. Shi.** A Statistical Fault Coverage Metrics for Realistic Path Delay Faults. *Proceedings of the 22nd IEEE VLSI Test Symposium* (*VTS*'04), 2004, 37–42.

[8] **W. Qiu, D.M.H. Walker.** An Efficient Algorithm for Finding the K Longest Testable Paths Through Each Gate in a Combinational Circuit. *Proceedings of the IEEE International Test Conference*, *Charlotte, NC, Sept*. 2003, 592-601.

[9] **Y. Shao, I. Pomeranz, S.M. Reddy.** On Generating High Quality Tests for Transition Faults. *Proceedings of the* 11th *Asian Test Symposium* (*ATS*'02), 2002, 1-8.

[10] **M. Sharma, J.H. Patel.** Testing of Critical Paths for Delay faults. *Proceedings of the IEEE International Test Conference*, Baltimore, *MD, Nov*. 2001, 634-641.

[11] **M. Sharma, J.H. Patel.** Finding a Small Set of Longest Testable Paths that Cover Every Gate. *Proceedings of the IEEE International Test Conference*, *Baltimore, MD, Oct*. 2002, 974-982.

[12] **G.L. Smith.** Model for Delay Faults Based Upon Paths. *Proceedings of the IEEE International Test Conference*, *Philadelphia, PA, Oct.* 1985, 342-349.

[13] **J. Waicukauski, E. Lindbloom, B. K. Rosen, V.S. Iyengar.** Transition Fault Simulation. *IEEE Design & Test of Computers*, *Vol*. 4, *No*. 2, *Apr*. 1987, 32-38.

[14] **T.W. Williams, B. Underwood, M.R. Mercer.** The Interdependence Between Delay-Optimization of Synthesized Networks and Testing, *ACM/IEEE Design Automation Conf.*, *San Francisco, CA, June* 1991, 87-92.

[15] **K. Yang, K.-T. Cheng, L.-C. Wang.** TranGen: A SAT-Based ATPG for Path-Oriented Transition Faults. *Proceedings of the ASP-DAC*, 27-30 *January* 2004, 92-97.

Received June 2009.

DOI: 10.5755/j01.itc.38.4.12076