An Efficient Diagnosis Method using Pattern Comparison

Hyungjun Cho
Electrical & Electronic Engineering Department
Yonsei University
Seoul, Korea
chj0937@soc.yonsei.ac.kr

Joohwan Lee
Electrical & Electronic Engineering Department
Yonsei University
Seoul, Korea
eldsich@soc.yonsei.ac.kr

Yoseop Lim
Electrical & Electronic Engineering Department
Yonsei University
Seoul, Korea
yoseop@soc.yonsei.ac.kr

Sungho Kang
Electrical & Electronic Engineering Department
Yonsei University
Seoul, Korea
shkang@yonsei.ac.kr

Abstract - *In this paper, we present an efficient diagnosis method using pattern comparison for reducing fault candidate lists. Critical path tracing determines fault candidate lists detected by a set of test patterns using a backtracing algorithm starting at the primary outputs of a circuit. Proposed algorithm reduces fault candidate lists by comparing failed patterns with good patterns during critical path tracing process. The proposed algorithm increases the simulation speed and the accuracy of diagnosis, and is applicable to both combinational circuits and sequential circuits. Experimental results on ISCAS'85 benchmark circuits show fault candidates lists is reduced more than by previous diagnosis methods.*

Keywords: *pattern comparison, critical path tracing, fault candidate, diagnosis, backtracing.*

1. Introduction

By development of semi-conductor process technology, complexity of circuit is increasing day after day. According as all systems come to single chip, type of defects and the number of defects is increasing. According as the type and the number of defects increased with this, it became important that repair finding the cause and the type of defects. Fault diagnosis is the process that deduces the location of defects which caused failures.

Fault diagnosis [1],[2],[3],[4],[5] based on fault dictionaries can be characterized as a cause-effect analysis [1],[6] that starts with possible causes and determines their corresponding effects. A second type of approach, employed by several diagnosis methods, relies on an effect-cause analysis [5],[7] in which the effect (the actual response obtained from the unit under test) is processed to determine its possible causes.

An alternative to fault simulation, referred to as critical path tracing [3], determines faults detected by a set of tests using a backtracing algorithm starting at the primary outputs of a circuit. Critical path tracing is faster and requires less memory than conventional fault simulation. Because critical path tracing starts at the failed primary outputs of a circuit through the critical paths, we can reduce the number of suspected fault candidates. So, we can reduce the total time of fault diagnosis. As we reduce all suspected fault candidates of the circuit to more precisely suspected fault candidates, we can also obtain the exact fault diagnosis resolution [8].

However, because the method through critical path tracing is thought from all targets to fault candidates in critical path, if we perform fault diagnosis on all fault candidates which are created through critical path tracing, the time cost is much greater yet. In this paper, we present the method to reduce the number of fault candidates which are obtained from critical path tracing through comparison the logic value of failing patterns with the logic value of good patterns. From the simulation results, we can confirm that fault diagnosis time is decreased in proportion to fault candidate decrease. We propose that our algorithm can reduce the number of fault candidates from the simulation results.

2. A Pattern comparison algorithm for reducing fault candidates

2.3 The order of fault diagnosis

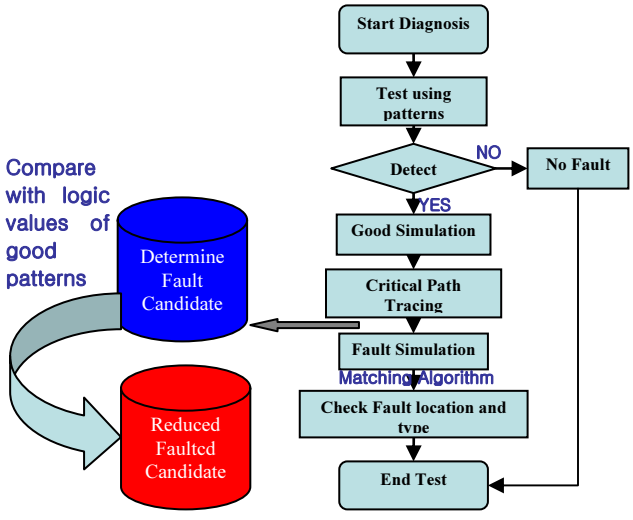

Figure 1. Order of fault diagnosis

Figure 1 shows the order of fault diagnosis. First of all, if a test detects faults in circuit, a true-value simulation is performed. And critical path tracing determines fault candidate lists. We performed the fault simulation with these lists. Then we were then able to determine the location of faults and the type of faults.

In like manner, the number of fault candidates determine the time of fault simulation [8]. Therefore, any reduction of the number of fault candidates decreases the fault simulation time and improves fault diagnosis resolutions. Our pattern comparison algorithm presented here is designed to reduce the number of fault candidates.

2.2 A Matching Algorithm

Both static fault diagnosis using a fault dictionary and dynamic fault diagnosis with fault simulation can determine the location and the type of real faults by comparing result values of experiments through fault responses of circuits including real faults.

This comparison and analysis determines the resolution of the fault diagnosis result. A matching algorithm determines the process of the comparison and the analysis. The performance of a matching algorithm determines the accuracy of the fault diagnosis results. Therefore, an accurate matching algorithm is very important in the whole fault diagnosis process.

In this paper, an efficient matching algorithm [2] has application to fault diagnosis that can use both static and dynamic fault diagnoses. This matching algorithm can deal flexibly with each different circuit while considering the number of the outputs. Using this matching algorithm, we can obtain accurate results of fault diagnosis.

2.3 A pattern comparison algorithm

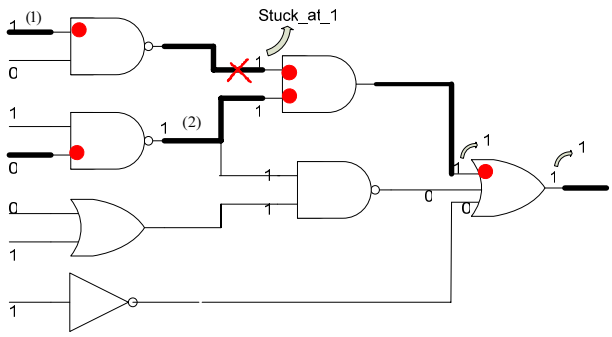

Figure 2. Example CPT from a good pattern

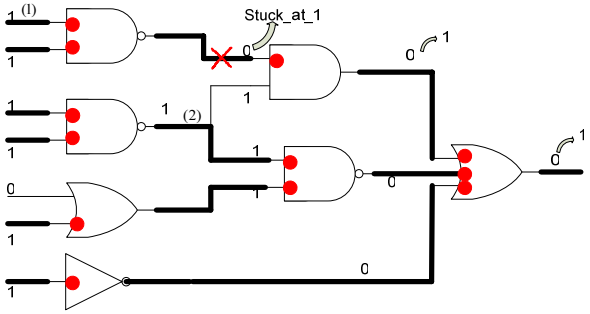

Figure 3. Example CPT from a failed pattern

The proposed algorithm is designed to reduce fault candidate lists by eliminating needless fault candidates through a comparison of logic values of good patterns with logic values of failed patterns. If a line has sensitivity in good patterns and it has also the same logic value as the logic value in the failed pattern, then the fault of the line must be found in the primary output of the circuit in good patterns. Therefore, if it is a real fault, it must not have the same values through a comparison with sensitive inputs through good patterns. The proposed algorithm can prune unnecessary fault candidates for this reason.

As an example, candidate (1) and candidate (2) are able to be pruned in Figures 2 and 3, because (1) and (2) have the same logic values (through CPT in a good pattern and through CPT in a failed pattern). If there is a stuck-at-0 fault in (1) or (2), the fault should have transmitted to primary output in Figure 2. Therefore, if a fault candidate has sensitivity and the same value both in a good pattern and in a failed pattern, then that fault candidate can be pruned.

Figures 4, 5, and 6 are pseudo-codes for the proposed algorithm. First of all, we must know the failed patterns that faults present to the primary outputs. We must also determine which faults represent what cycles and what primary outputs. According to the number of good patterns that are compared with a failed pattern, the number of reduced fault candidates differs. Therefore, we must determine the number of good patterns to compare. In this paper, we compared all good patterns with each failed pattern. In the following steps (1) ~ (4), we explain these pseudo-codes seen in Figures 4, 5, and 6.

(1) First of all, we should know all failed outputs in all failed patterns. And in a good pattern, we store logic values and critical paths (in specific memory spaces) that are obtained through critical path tracing in a good pattern.

(2) Like (1), we store logic values and critical paths (in specific memory spaces) that are obtained through critical path tracing in all good patterns.

(3) We store logic values and critical paths in specific memory spaces that are obtained through a critical path tracing in each failed pattern.

(4) If any fault candidates have same logic values and they are sensitive inputs in both good patterns and failed patterns, then we can eliminate these fault candidates.

```
for failed output (FPO) in a good pattern
{
    While (good_sensitivity input ≠∅ )
        Critical path tracing(FPO)
    for every good_sensitivity inputs
        value[i] = logic_value
}
```

Figure 4. Critical path tracing using the good patterns

```
for failed output (FPO) in a failing pattern
{
    While (failed_sensitivity input ≠∅ )
        Critical path tracing(FPO)
    for every failed_sensitivity inputs
        value[j] = logic_value
}
```

Figure 5. Critical path tracing using the failed patterns

```
for every failed_sensitive inputs
{
    if (value[i] = value[j] & good_sensitivity)
        eliminate this input
}
```

Figure 6. Eliminate unnecessary fault candidates through pattern comparisons

3. A fault masking problem

If there is fault masking in the good patterns, then the fault is not presented to the primary outputs, which may produce wrong results. In this paper, to prevent this problem, we eliminated unnecessary fault candidates when a gate's fault candidate is only one sensitive input of the gate in good patterns.

4. Simulation results

Table 1. Results for c7552 failing circuits

c7552 failing Circuits	Candidates (without CPT)	Candidates (with CPT)	Candidates (reduced)	Diagnosis Time (with CPT)	Diagnosis Time (reduced)
1	5054	1042	**790**	7.66	**6.21**
2	5054	495	**362**	4.1	**3.75**
3	5054	830	**559**	6.41	**5.62**
4	5054	190	**135**	1.71	**1.82**
5	5054	602	**384**	4.53	**3.38**
6	5054	885	**619**	6.77	**5.46**
7	5054	342	**248**	2.91	**2.75**
8	5054	708	**438**	5.47	**4.01**
9	5054	831	**558**	6.16	**4.84**
10	5054	1026	**672**	7.89	**5.65**
11	5054	221	**159**	1.94	**2.03**
12	5054	1433	**1102**	10.6	**8.88**
13	5054	765	**493**	5.93	**4.66**
14	5054	934	**640**	7.22	**5.72**
15	5054	1530	**1264**	11.56	**10.91**
16	5054	726	**502**	5.54	**4.78**
17	5054	1117	**800**	8.16	**6.32**
18	5054	1157	**873**	8.53	**7.17**
19	5054	1316	**863**	10.01	**7.29**
20	5054	821	**531**	6.27	**4.7**
21	5054	909	**613**	7	**5.45**
22	5054	1149	**914**	8.66	**7.78**
23	5054	471	**302**	3.8	**3.34**
24	5054	347	**202**	2.9	**2.33**
25	5054	367	**235**	3.04	**2.54**
26	5054	1784	**1407**	13.56	**12.08**
27	5054	872	**597**	6.47	**4.85**
28	5054	695	**451**	5.31	**4.17**
29	5054	235	**135**	2.15	**1.94**
30	5054	582	**363**	4.74	**3.84**
Average	**5054**	**812**	**573**	**6.23**	**5.14**

Table 2. Average results of candidates for each circuit

Circuit	Total candidates	CPT candidates (a)	Reduced candidates (b)	(b) / (a)
c1355	1219	765	**533**	**0.69**
c2670	1742	333	**243**	**0.73**
c5315	3978	540	**363**	**0.67**
c6288	8875	4441	**2953**	**0.66**
c7552	5054	813	**574**	**0.71**
Average				**0.69**

Table 3. Average results of total times for each circuit

Circuit	(a) Total time (through CPT)	(b) Reduced time	(b) / (a)
c1355	0.99	0.78	**0.79**
c2670	1.76	1.49	**0.85**
c5315	3.87	3.32	**0.86**
c6288	16.65	12.85	**0.77**
c7552	6.23	5.14	**0.83**
Average			**0.82**

Table 4. Comparison of candidates with the number of compared good patterns

Circuit	(a) Candidate (1:1)	(b) Candidate (1:2)	(c) Candidate (1:all)	(c) / (a)	(c) / (b)
c5315	491	447	**363**	**0.74**	**0.81**
c6288	4164	3836	**2953**	**0.71**	**0.77**
c7552	745	698	**574**	**0.77**	**0.82**
Average				**0.74**	**0.80**

Table 5. Comparison of total diagnosis time with the number of compared good patterns

Circuit	(a) Total time (1:1)	(b) Total time (1:2)	(c) Total time (1:all)	(c) / (a)	(c) / (b)
c5315	3.87	3.69	**3.32**	**0.86**	**0.90**
c6288	16.36	15.42	**12.85**	**0.79**	**0.83**
c7552	6.15	5.99	**5.14**	**0.84**	**0.86**
Average				**0.83**	**0.86**

In this paper, experiments were performed with huge combinational ISCAS'85 benchmark circuits (c1355, c2670, c5315, c6288, c7552). In these experiments, we used 30 failing circuits, each failing circuit having two faults. We compared with all good patterns about each failed pattern, and we can diagnose with precision using MAID [acronym's meaning or different new name] (a diagnosis tool developed in our lab.). MAID is software using matching algorithm.

Table 1 shows simulation results of testing a c7552 circuit for the number of fault candidates without a critical path tracing, the number of fault candidates and the total diagnosis time with critical path tracing, and the number of fault candidates and the total diagnosis time with pattern comparison algorithm. Table 2 shows the proposed algorithm reduces the number of fault candidates and the total diagnosis time. The reduction of fault candidates decreases the total diagnosis time. The reduced total diagnosis time is shown in Table 3. In Tables 2 and 3, we find the number of fault candidates is reduced by about 30% and the total diagnosis time is reduced by about 20%.

In this paper, experiments were conducted with all good patterns. Tables 4 and 5 show comparison results for the number of good patterns compared with each failed pattern. The tables show that if the number of compared good patterns is increased, then the number of eliminated fault candidates is also directly increased.

A good pattern can eliminate more needless fault candidates in sequential circuits than in combinational circuits. Therefore, an application of the pattern comparison algorithm to sequential circuits is expected to show even better results.

5. Conclusions

Experiments were performed through the critical path tracing approach. From the results of the critical path tracing (that is, the backtracing method), we determined fault candidate lists, and from the results of fault simulation with these fault candidates, we confirmed the fault location and type.

The number of fault candidates determined the total fault simulation time. To decrease the total fault diagnosis time, it is essential to reduce the number of fault candidates. Therefore, we have presented a method to eliminate the unnecessary fault candidates through comparison logic values of good patterns with logic values of failed patterns if fault candidates have sensitivities in both good patterns and failed patterns. In conclusion, we show that the proposed algorithm reduces the total fault diagnosis time by about 20%. Even though our experiments were performed only on combinational circuits, the pattern comparison algorithm should also prove applicable to sequential circuits.

References

[1] S. D. Millman, E. J. McCluskey and J. M. Acken, "Diagnosing CMOS Bridging Faults with Stuck-At-Fault Dictionaries", Proc. of International Test Conference, pp. 860-870, 1990.

[2] Joohwan Lee, Yoseop Lim, Hyungjun Cho, Sungho Kang, "An Efficient matching Algorithm using the Number of Primary Outputs for Fault Diagnosis", 2005 SOC Design Conference CD, pp 295-298, 2005.

[3] Hyungjun Cho, Joohwan Lee, Yoseop Lim, Sungho Kang, ,"An Effective fault diagnosis using critical path tracing", Proc. of Korea Test Conference, pp 8-12, 2005.

[4] M. Abramovici, "A Maximal Resolution Guided-Probe Testing Algorithm", Proc. of Design Automation Conference, pp. 189-195, 1989.

[5] M. Marzouki, J. Laurent and B. Courtois, "Coupling Electron-Beam Probing with Knowledge-Based Fault Localization", Proc. of International Test Conference, pp. 238-247, 1991.

[6] I. Pomeranz and S. M. Reddy, "On Dictionary-Based Fault Location in Digital Logic Circuits" IEEE Transactions on Computers, pp. 48-59, 1997.

[7] S. Venkataraman, Ismed Hartanto and W.K. Fuchs, "Dynamic Diagnosis of Sequential Circuits Based on Stuck-at Faults", Proc. of VLSI Test Symposium, pp. 198-203, 1996.

[8] Jiang Brandon Liu, "Incremental Fault Diagnosis", IEEE Transactions on Computers, pp.240-251, 2005