VLSI Testing Final Project Topic #2 Diagnosis

Yu-Min Li*, Yu-Tsung Wu* and Zhe-Jia Liang†
*Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan
†Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan
{r09943094, r09943111, b06901161}@ntu.edu.tw

***Abstract*—This is a report for VLSI testing final project. The topic is stuck-at fault diagnosis. We implement a fail log generation function and design a stuck-at fault diagnosis tool. The experimental results of diagnosis show that the accuracy and resolution are acceptable in nine benchmark circuits. The average diagnosis accuracy of each circuit are not less than 0.86. The average diagnosis resolution of each circuit are not more than 4.05.**

I. Introduction

Diagnosis is a process that find out some candidate locations of defects. The good accuracy and resolution of diagnosis help manufactures improve circuit yield and quality. Single fault diagnosis method is mature and has linear-time complexity solution. But the existence of multiple defects increase the difficulty of diagnosis since different defects might affect the expression of each other. In this project, we design a diagnosis tool which can find a candidate locations on a logic circuit with multiple stuck-at faults.

The methodology we proposed has two steps. First, using the single fault diagnosis method. If there is any fault response consistent to the input fail log. We then report that fault as our suspect. Else we enter the second step, multiple faults diagnosis. Multiple faults diagnosis select the initial candidate faults by path-tracing. Next, insert all candidate faults and do some simulation for each test pattern to eliminate some candidate faults. With the rest of candidate faults, we do scoring over each fault based on the simulation fault response and the test response. Pick the faults with the highest score into final suspect fault set iteratively. In this project, we set the iterative time equal to two from the experiment.

The rest of this paper are organized as the follows. Section II gives background of diagnosis process.Section III shows our proposed methods including single fault diagnosis and multiple faults diagnosis. The experimental results are showed in section IV. Some discussions are given in section V. Finally, we make conclusions in section VI.

II. Background

When a chip is failed in testing procedure, we would like to find out what is the true defects of chip failure, this so called diagnosis. When we give the information of circuit under diagnosis(CUD) to a diagnosis tool, it will report the possible faults of the CUD, called suspect faults. We usually consider the accuracy and resolution as the metrics of a diagnosis tool. Accuracy means the rate that number of true defects reported by the diagnosis tool over the total number of true defects. Perfect accuracy is 1, the higher the better. Resolution is the rate that total number of fault reported by the diagnosis tool over number of true defects reported by the diagnosis tool. Perfect resolution is 1, the lower the better.

In circuit diagnosis, we usually think the defects can be as simple as possible. That is, single fault is more likely than double faults. However, it is the fact that multiple faults possibly occur. In this project, we consider the stuck-at fault model and multiple faults assumption. The difficulty of multiple-fault diagnosis is that the number of multiple-fault sets grows exponentially. In addition, masking and reinforcement effects are also the problems of multiple-fault diagnosis. Without other circuit information, it is challenging to diagnose circuit with multiple faults.

III. Proposed Methods

A. Overview

In logic circuit diagnosis, we assume the situation is as easier as possible. First, we assume our CUD has only one fault. In this assumption, we pick out the circuit with only one fault. If the faulty response perfectly match to the fail logs, we believe the CUD is a single-fault circuit. As a result, we do multiple fault diagnosis only when there are some mismatches between each single fault's faulty response and the fail log.

The overview of our methodology is shown in Figure 1.

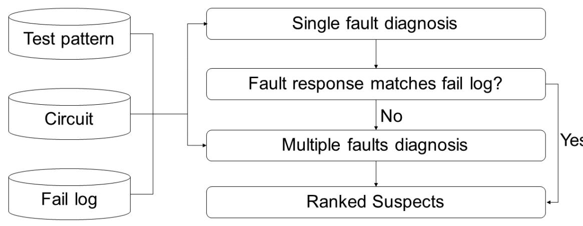

Fig. 1. The overview of proposed methodology

B. Single Fault Diagnosis

The flow chart for single fault diagnosis is shown as Figure 2. Input information contains test patterns, circuit structure and fail log.

First, we read the failing bit and corresponding pattern. Next, we trace back the circuit and construct the fan-in cone of

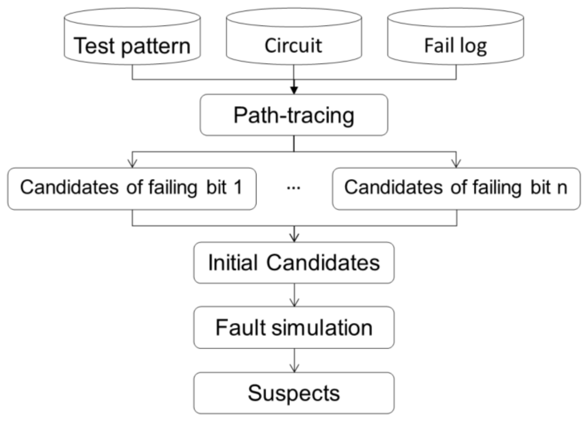

Fig. 2. The flow for our single fault diagnosis

the failing bit. Then, we choose the candidates of this failing bit. The method of choosing candidates is as follow [1]. In the beginning, We perform a simulation for fault-free circuit. After that, we determine whether the candidate to be chosen or not based on the gate type and fault-free wire value. For example, if a 2-input AND gate with output value 1, both two inputs are chosen as candidates. Because flipping any of the two wires will result in flipping output value, both two wires need to be chosen as candidates. For other gate types, we choose the candidates by the same rule. We perform choosing candidates operation on each failing bit by method mentioned above. After choosing the candidates, we get the candidate set of each failing bit. We take the intersection of all candidate sets. If the faulty behavior caused by only one fault, the fault must exist in all the candidate sets. Because the single fault assumption, we can easily select the candidates if the CUD is a single-fault circuit.

We choose the intersection of all candidate sets as the initial candidate set. However, it is not enough to take the set as the final suspect faults. We now perform single-fault simulation for all initial candidate faults. Please note that we have the original input patterns information of fail logs. The fail log information contains only failing bit on corresponding pattern. That is, we definitely know that some of the patterns with good outputs and others with faulty outputs. If the faulty behaviors are caused by only one fault, the faulty response of CUD will precisely match with fail log based on input patterns. After performing fault simulation on each initial candidate fault, the fault which perfectly match fail logs is chosen as suspect fault. If there exist at least one fault in suspect fault set, we can say that the CUD is a single-fault circuit. Please note that the root cause of the CUD may have multiple faults. We still say that the CUD is a single-fault circuit if the suspect fault set is not empty. The suspect fault set may contain many faults due to the limitation of input patterns and fail log information. Hence, we choose any one of suspect faults as the final suspect fault because we cannot distinguish which is the real fault location.

For those CUD with non-empty suspect fault set, we can say that it is a single-fault circuit. The advantage of single fault diagnosis procedure is small run time. We can quickly diagnosis the CUDs with single fault. However, most of CUDs with multiple faults cannot be found any faults to explain whole fail log. Only a few multiple-fault CUD can be explained by single fault. Obviously, single fault diagnosis is not robust enough to handle all CUDs. Therefore, we need another diagnosis method to enhance our work.

C. *Multiple Faults Diagnosis*

The flow chart for multiple fault diagnosis is shown as Figure 3, and the pseudo code for DSIE algorithm [2] is given in Figure 4.

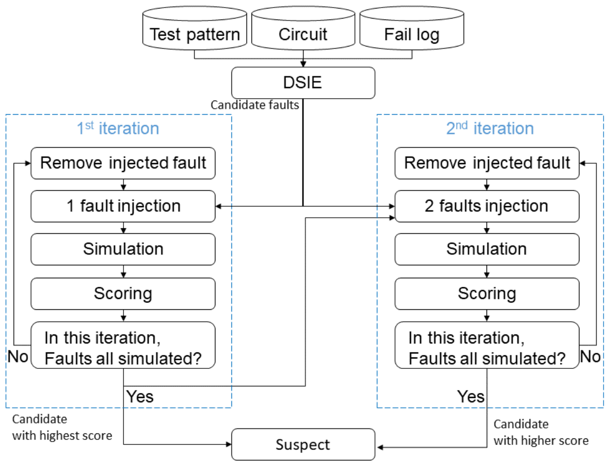

Fig. 3. The flow of our multiple fault diagnosis

Algorithm 1 DSIE Algorithm

```
for each failing test pattern $t_k$ do
  Path-trace to identify potential defect sites ($D_k$) implicated by $t_k$
end for
$D$ = union of all $D_k$s
while first iteration or at least one site eliminated in last iteration do
  for each failing test pattern $t_k$ do
    Simulate $t_k$
    Inject the unknown value $X$ at each potential defect site in $D$
    Propagate all $X$ values to the outputs
    Assign fault-free logic values to all passing outputs
    $Queue$ = all lines with values changing from $X$ to $v \in \{0, 1\}$
    while $Queue \neq \emptyset$ do
      $s_i$ = line popped from $Queue$
      Forward imply from $s_i$
      $FI_i$ = lines with values $X$ and implied to $v \in \{0, 1\}$
      $FA_i = FI_i - D$
      Lines in $FA_i$ are assigned to implied values
      $Queue = Queue \cup FA_i$
      if $s_i \notin D$ then
        Backward imply from $s_i$
        $BI_i$ = lines with values $X$ and implied to $v \in \{0, 1\}$
        Lines in $BI_i$ are assigned to implied values
        $Queue = Queue \cup BI_i$
        $BP_i$ = sites implicated by path-tracing only through $BI_i$
        $D_k = D_k - BI_i - BP_i$
        $D$ = union of all $D_k$s
      end if
    end while
  end for
end while
```

Fig. 4. The DSIE algorithm [2]

The purpose for using DSIE algorithm is to generate candidate list. First it would do path tracing from all failing outputs of all test patterns to get an initial candidate fault list. Second, it will inject all candidate faults to the CUD with value X to represent the all possible interactions among all candidate faults [2]. After all faults injected, for each test pattern, it would do fault simulation and compare the result with the fault-free simulation. If there is any observable point that passed in the fail log but has value X after fault simulation, it means that there exist some redundant candidates that can be eliminated. The reason is that even when considering all possible interactions, the fault effects still should not propagate to such observable point, which implies there too much injected faults and there exist redundant candidate faults. Then DSIE algorithm will use backward implication from such observable points to see if there are any wire that cannot be influenced by the fault effects and hence can eliminate candidate faults.

DSIE algorithm will run until there is no candidates can be removed; however, the number of candidates could be still very large. Although some groups eliminate more candidates using fault propagation and fault activation conditions, such method do not consider the reinforcement and the masking problem when dealing with multiple faults. On the other hand, although there are some algorithms can further eliminate candidates and deal well with multiple faults, they are too complicated for us to implement. Hence we just using DSIE algorithm alone.

After DSIE algorithm, we now have the candidate faults list. Before entering the first iteration, we need to explain the score used in our multiple fault diagnosis methodology first. There are two scores: Match_Sum and FM_Det [3]. Match_Sum equals to the number of matched observable points between the fail log and the fault simulation result, which is, in other words, the summation of TPSP and TFSF observable points. FM_Det is the number of test pattern that test result and simulate result are fully matched. If a fault's FM_Det equals to the number of test patterns, it means that that fault can perfectly explain the fail log. For example, if the CUD has only single fault and has i test patterns, there should be at least one fault that whose FM_Det score is i.

In this project, FM_Det is more important than Match_Sum, and hence if we can find a combination of faults that generate the highest FM_Det score (that is, the number of test patterns,) we believe it is the correct answer. However, there are too many combinations of faults. To solve this problem, we do some research on the fail log and discover one interesting thing.

First we define true suspects as those faults injected by the TA. By observing the fail log, we discover that if a fault f belongs to the true suspects, no matter how many true suspects there are, f's score is still very high even when we only inject f. As a result, we do not consider too many combinations of faults but just use two iterations to select faults greedily.

In the first iteration, we will inject a fault that has not been simulated in current iteration, and calculate its score. After all candidate faults have been injected and simulated, we will find a fault with highest FM_Det score; if there are more than one fault with the same FM_Det score, we will choose those have highest Match_Sum score and the chosen faults are directly output as suspects with the same ranking.

In the second iteration, we inject two faults at a time. One is the chosen fault in the first iteration, and the other is a fault that has not been simulated in the second iteration. Next step is to simulate the circuit and judge the interaction between two faults by scoring. After all faults are simulated, we sort the candidate faults by their score and the faults with higher scores will be output as suspects.

Since we choose suspects just by their scores, this is a greedy methodology. If the fault chose in the first iteration is indeed one of the true suspect, it should be easier to find other true suspects in the second iteration since the multiple faults effect are correctly considered. On the other hand, because higher score represents higher similarity to the true suspect, the diagnosis result would not be ruined when the chosen fault in the first iteration does not belong to true suspects.

IV. Experimental Results

In our result, we will output only one suspect if we can use single fault to explain the fail log. If we output more than one suspect, it means that we need multiple faults to explain the fail log and we will use a variable named groupID to classify faults by their score. Faults with the same groupIDs will have the same score and hence can be grouped together. In this project, our groupID is only ranged from 1 to 5; as a result, all of our output suspects belong to the top 5 faults and could be counted as correctly diagnosed faults.

Below is the average result. The first column is the name of the CUDs, the other columns are accuracy, resolution, and run time respectively. From table I, it shows that when circuit size grows, the diagnosis run time increases as well.

TABLE I
Average Result

Circuit number	Diagnosis accuracy	Diagnosis resolution	Run time.
C17	1	1.185	0
C432	1	1.05	0.015
C499	0.935	1.725	5.965
C880	0.9875	4.05	1.87
C1355	1	1	2.265
C2670	0.928	2.62	79.3
C3540	1	1.775	22.38
C6288	0.8625	1.68	250.53
C7552	0.908	2.2	1194.84

Below we will show our results for each circuit. The first column is the index of fail log, the second column is our diagnosis accuracy, the third column is our diagnosis resolution and the fourth column is named as suspect difference and will list the cases which generate the same fail logs by different suspects.

Our results shows that the rum time for single fault diagnosis is very fast, and is very slow for multiple fault diagnosis. Besides, it's interesting that there are many multiple-fault effects can be explained by less faults; this will be discussed in the next section.

TABLE II
C17 RESULT

Id	Acc.	Res.	RT	Suspects Diff.
1	1	1	0	
2	1	1	0	
3	1	1	0	
4	1	1	0	
5	1	1	0	
6	1	1	0	
7	1	1	0	
8	1	1	0	
9	1	1	0	
10	1	1	0	
11	1	1.5	0	22GAT g6 GO SA1 7GAT dummy_gate5
12	1	1	0	
13	1	1	0	23GAT g5 GO SA1
14	1	1	0	11GAT g1 GO SA1 16GAT g5 GI SA1
15	1	1	0	16GAT g4 GO SA1
16	1	1	0	3GAT dummy_gate3 GO SA1 19GAT g3 GO SA1
17	1	1	0	16GAT g4 GO SA0
18	1	2	0	11GAT g1 GO SA0+3GAT dummy_gate3 GO SA0
19	1	1	0	16GAT g4 GO SA0
20	1	3.2	0	

TABLE III
C432 RESULT

Id	Acc.	Res.	RT	Suspects Diff.
1	1	1	0.1	223GAT g49 GO SA0
2	1	1	0	
3	1	1	0	108GAT g21 GI SA1
4	1	1	0	
5	1	1	0	168GAT g46 GI SA1
6	1	1	0	223GAT g49 GO SA0
7	1	1	0	223GAT g49 GO SA0
8	1	1	0	223GAT g49 GO SA0
9	1	1	0	95GAT g24 GI SA1
10	1	1	0	150GAT g2 GO SA1
11	1	1	0	4GAT g45 GI SA1
12	1	1	0	108GAT g21 GI SA1
13	1	1	0	168GAT g46 GI SA1
14	1	1	0	168GAT g46 GI SA1
15	1	1	0	146GAT g4 GO SA1
16	1	1	0	223GAT g49 GO SA1
17	1	2	0.2	69GAT dummy_gate22 GO SA1 17GAT g42 GI SA1
18	1	1	0	223GAT g49 GO SA0
19	1	1	0	171GAT g46 GI SA1
20	1	1	0	223GAT g49 GO SA0

V. Discussion

For single fault diagnosis, we can get good accuracy in a short time. The resolution may be bad since there are too many faults that have the same fault effect to the test pattern. For multiple fault diagnosis, the resolution also have the same problem. Also, we encounter many cases that our suspects are different from the true suspects but have the same or very similar fault effect on the CUDs. To solve those problems, we will need more patterns to distinguish each fault's behavior. If possible, insert more observation points can also help us to find difference between those faults with the same score.

TABLE IV
C499 RESULT

Id	Acc.	Res.	RT	Suspects Diff.
1	1	1	0.1	
2	1	1	0.5	
3	1	1	0.1	
4	1	1	0.5	
5	1	1	1	
6	1	1	0	
7	1	1	0.1	
8	1	1	0.7	
9	1	2	8.2	ID13 g371 GI SA1 ID12* g373 GI SA1
10	0.5	5	12.1	
11	1	1.5	8.4	
12	1	1.5	8.5	
13	1	1.5	10	
14	1	3.33	9	
15	1	1.5	8.7	OD29 g321 GO SA0 ID8* g79 GI SA1
16	1	2	10.5	
17	1	2	9.7	
18	1	1.5	10.2	
19	0.6	2.33	10.9	
20	0.6	2.33	10.1	

TABLE V
C880 RESULT

Id	Acc.	Res.	RT	Suspects Diff.
1	1	1	0	
2	1	1	0.3	
3	1	1	0	
4	1	1	0.2	794GAT g515 GO SA1
5	1	1	0	
6	1	1	0	687GAT g263 GI SA1
7	1	1	0	827GAT g331 GO SA1
8	1	1	0	
9	1	1	0.1	393GAT g106 GO SA1
10	1	1	0	879GAT g382 GO SA1
11	1	1	0	845GAT g353 GO SA1
12	1	1	5.7	
13	1	1	0	448GAT g120 GO SA1
14	1	2	3.4	270GAT* g77 GI SA1 722GAT g236 GO SA1
15	1	4.33	7.2	
16	1	1.5	5.9	625GAT g218 GO SA1 51GAT g51 GI SA1
17	1	1	0.8	635GAT g239 GI SA1
18	0.75	55.67	4.9	
19	1	2	6.1	565GAT g195 GO SA0 830GAT* g536 GO SA1
20	1	1.5	2.8	446GAT g124 GO SA1 727GAT g235 GO SA1

As for the multiple fault diagnosis, out methodology seems to be nice (good resolution and reasonable accuracy,) but it has a big problem on time complexity. The time would be as follow:

$$(\#\ of\ patterns) \times (\#\ of\ candidates) \times (\#\ of\ iterations)$$

TABLE VI
C1355 RESULT

Id	Acc.	Res.	RT	Suspects Diff.
1	1	1	0.5	233GAT dummy_gate41 GO SA0
2	1	1	0.2	860GAT g294 GO SA0
3	1	1	0.2	1326GAT g535 GO SA0
4	1	1	0.3	
5	1	1	0.5	233GAT dummy_gate41 GO SA0
6	1	1	0	
7	1	1	0.2	860GAT g294 GO SA0
8	1	1	0.1	773GAT g263 GO SA1
9	1	1	0	1060GAT g385 GO SA1
10	1	1	0.2	860GAT g294 GO SA0
11	1	1	1.2	873GAT g293 GO SA0
12	1	1	12.1	
13	1	1	0.3	925GAT g292 GO SA0
14	1	1	13.2	996GAT g383 GI SA1 435GAT g130 GO SA1
15	1	1	14.7	1001GAT g372 GI SA1 43GAT dummy_gate7 GO SA1
16	1	1	0	1026GAT g357 GI SA1
17	1	1	0.1	43GAT dummy_gate7 GO SA1
18	1	1	0.5	233GAT dummy_gate41 GO SA0
19	1	1	0.5	492GAT g129 GO SA1
20	1	1	0.5	233GAT dummy_gate41 GO SA0

TABLE VII
C2670 RESULT

Id	Acc.	Res.	RT	Suspects Diff.
1	1	1	1.3	
2	1	1	0.4	
3	1	1	4	1174 g1775 GO SA0
4	1	1	0.2	
5	1	1	8.4	1098 g1744 GO SA0
6	1	1	0.1	
7	1	1	5.4	
8	1	1	4.5	
9	1	1	53.4	
10	1	1	170.4	
11	1	1.5	172.6	
12	1	1.5	142.3	
13	1	3	140.8	
14	1	1.5	101.3	896 g1251 GI SA1 1125 g885 GO SA0
15	0.67	3.5	120	
16	1	1.5	81.1	16 g100 GI SA1 67 dummy_gate51 GO SA1
17	0.75	23	99.8	
18	0.75	2	150.7	
19	0.6	2.67	164.1	
20	0.8	2.25	165.2	

TABLE VIII
C3540 RESULT

Id	Acc.	Res.	RT	Suspects Diff.
1	1	1	0.6	
2	1	1	2.8	
3	1	1	1.7	
4	1	1	6.4	
5	1	1	0.5	2828 g1591 GO SA0
6	1	1	1	2903 g1634 GI SA1
7	1	1	0.7	2302 g187 GO SA0
8	1	1	0.5	353 g356 GO SA0
9	1	1	0.3	41 dummy_gate5 GO SA1
10	1	1	2.9	2236 g1744 GO SA1
11	1	1	1.5	369 g1272 GO SA0
12	1	1	0.9	402 g1669 GO SA0
13	1	1	0.2	704 g158 GI SA1
14	1	1	1.9	2043 g1165 GI SA1
15	1	1	0.2	169 dummy_gate23 GO SA0
16	1	1	1	2062 g814 GI SA1
17	1	1	4	3131 g1135 GO SA1
18	1	1	232.1	58 dummy_gate8 GO SA0 736 g283 GI SA1
19	1	14	86.3	2983 g310 GO SA0 1791 g412 GI SA1
20	1	3.5	102.1	898 g116 GI SA1 294 dummy_gate40 GO SA1

TABLE IX
C6288 RESULT

Id	Acc.	Res.	RT	Suspects Diff.
1	1	1	5.6	
2	1	1	1.8	
3	1	1	4	
4	1	1	1.9	4601GAT g1732 GI SA1
5	1	1	5.1	
6	1	1	9.4	
7	1	1	1.3	
8	1	1	1.4	
9	1	2	304.9	
10	1	1	409.3	
11	1	1	326.2	
12	1	1.5	437	
13	1	1	424.2	
14	0.67	2.5	437.8	
15	0.67	2.5	438.9	
16	0.67	3	442.1	
17	0.5	3.5	434.3	
18	0.75	1.67	439.8	
19	0.6	2	446	
20	0.4	4	439.6	

Since it is very easy to have thousands of candidates, the total run time could be very long. Hence to improve the performance, it is necessary to further eliminate candidate size while considering the reinforcement and masking effect of multiple faults.

Besides, our multiple fault diagnosis is based on the observation: if a fault f belongs to the true suspects, its score would remain high even if only inject f. This observation is true when the reinforcement and masking effects of multiple faults are not very critical. Hence it is very possible to be incorrect as the number of defects increase. When the number of defects increased, the number of faults increased as well and the interaction between faults will be more complicated, which imply that reinforcement and masking occurs more frequently.

Last but not the least, the number of iterations is important as well. When there are more iterations, not only the run time will increased, the accuracy may also increased since there are more faults considered at a time. However, not every iteration will give us the true suspect; if the chosen fault is not true suspect, there must exist some mismatch. When the number of iterations is two, the mismatch effect is negligible, but the mismatch will accumulate; more iterations result in larger mismatch, which will finally cannot be ignored. Hence

TABLE X
C7552 RESULT

Id	Acc.	Res.	RT	Suspects Diff.
1	1	1	28.2	
2	1	1	1.2	
3	1	1	2.1	6041 g157 GO SA1
4	1	1	0.8	
5	1	1	1.9	
6	1	1	0.9	
7	1	1	0.7	
8	1	1	1.4	5428 g3632 GO SA0
9	1	1.5	880.4	
10	1	1	1199.5	
11	1	2	900.6	
12	1	1.5	1625.3	
13	1	1.5	953	
14	1	2	2376.5	
15	0.67	5	1978.9	
16	1	2.67	3072.7	
17	0.75	5.33	1944.5	
18	0.75	3.33	3450.3	
19	0.4	7.5	2077.3	
20	0.6	2.67	3400.6	

we need to pay attention to the number of iterations.

VI. Conclusion

When diagnosis, we encounter some problems. One is bad resolution since there are too many faults with the same score. Another is that our suspects are different from the true suspects since some multiple faults can be explained by single fault, or by multiple faults but with less faults. Although there are some methods to solve these problems such as insert observation points and add more test patterns, they are not objective of this project because their categories are design for testability and test pattern generation.

For single fault diagnosis, our accuracy and run time are both good. Although there are some cases with bad resolution, the overall resolution is still good. For multiple faults diagnosis, when the number of true suspects increased, if the fault effect cannot be explained by single fault, basically, the number of our output suspect would increased as well and the accuracy would decrease. And our the resolution is good with reasonable accuracy, but performance is limited by the run time. As a result, it's necessary to eliminate more candidate faults to improve the run time.

VII. Work Distribution

TABLE XI
WORK DISTRIBUTION

Member	function
Yu-Min Li	ssf_DFS single_diag diagnosis_map_insert ssf_diag_sim_a_vector diagnosis_fault_insert
Yu-Tsung Wu	gen_FailLog diagnosis_fault_insert diag_fsim_a_vector diag_fault_sim_evaluate diag_GI_get_faulty_wire diag_GI_non_control
Zhe-Jia Liang	DFS read_failLog diag_initial_candidate diag_remove_candidate diag_keep_remove_candidate diag_greedy_ranking diag_print_result generate_fault_list diag_ssf_print_result diag_faultsim_by_level diag_greedy_ranking_2nd_iter

References

[1] Miron Abramovici, Premachandran R. Menon, and David T. Miller. ”Critical path tracing-an alternative to fault simulation.” 20th Design Automation Conference Proceedings. IEEE, 1983.

[2] Xiaochun Yu, and Ronald DeShawn Blanton, ”Diagnosis of Integrated Circuits With Multiple Defects of Arbitrary Characteristics,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), vol. 29, no. 6, pp. 977-987, 2010.

[3] M.-J. Tasi, M. C.-T. Chao, J.-Y. Jou, and M.-C. Wu, “Multiple-fault diagnosis using faulty-region identification,” in Proc. Very-Large-Scale Integr. (VLSI) Test Symp., 2009, pp.123–128.