

Diagnose Compound Scan Chain and System Logic Defects

Yu Huang, Wu-Tung Cheng, Ruifeng Guo
Mentor Graphics Corporation, 8005 S.W. Boeckman Road, Wilsonville, OR 97070, USA

Will Hsu and Yuan-Shih Chen
Taiwan Semiconductor Manufacturing Company 8,
Li-Hsin Rd. 6, Hsinchu Science Park
Hsinchu, Taiwan 300-77, R. O. C.

Albert Man
Advanced Micro Devices Inc.
1 Commerce Valley Drive East
Markham, Ontario, Canada L3T 7X6

Abstract

Scan based diagnosis can be of great help to guide physical failure analysis, which is critical for the success of silicon debug and yield ramp up. In practice, diagnosis becomes more difficult if scan chain defects and system logic defects co-exist on one die, which are called compound defects in this paper. We first describe the challenges in diagnosing this type of compound defects. A novel diagnosis flow is proposed to diagnose the compound defects on scan chain and system logic. The diagnosis methodology was successfully applied in industrial designs.

1. Introduction

As today's VLSI designs approach nanometer process technologies, scan based diagnosis becomes inevitable to guide silicon debug, physical failure analysis (PFA) and yield ramp up. Scan based diagnosis usually consists of scan chain diagnosis and system logic diagnosis. Scan chain diagnosis targets the defects in scan chains, while system logic diagnosis targets the defects in the system logic outside the scan chains.

Plenty of prior work discussed how to perform scan chain and system logic diagnosis separately. Nevertheless, in system logic diagnosis, it was always assumed scan chains are working correctly. Similarly, in software based scan chain defect diagnosis, it was typically assumed that the system logic is defect-free [KUN94] [GUO01] [HUA03]. Although hardware based scan chain defect diagnosis methods [SCH92] [EDI95] [NAR97] [WU98] do not have to make any assumptions on the system logic, they are rarely applied in the real world due to the extra hardware overhead of the special scan architectures. In the rest of this paper we only focus on software based diagnosis.

In practice, however, we find it is not unusual that both the scan chain(s) and the system logic have defects on one die. This creates a conundrum of

circular logic - to diagnose the faulty scan chains requires verifying the functional logic first - in order to allow the scan chain to be used to test the functional logic. At some point getting caught in this circular thrasher does reach the point of diminishing returns [CRO05].

In case the compound defects exist on one die, the chain defects and system logic defects could be either correlated or uncorrelated. In [HUA04], a special defect called DACS was discussed. DACS stands for Defects that Affect Chain and System logic. It means one defect that could impact on both chain and system logic. However, the method proposed in [HUA04] cannot diagnose uncorrelated defects that impact chain and system logic independently. In addition, the method proposed in [HUA04] has to use per-shift-cycle simulation, which is very slow and impractical to be applied for very large designs. The method proposed in [YAN05] performs chain diagnosis by using a signal profiling technique. It can tolerate a small number of logic defects during chain diagnosis, but cannot diagnose system logic defects.

In this paper, we propose a novel diagnosis flow that tries to diagnose compound defects no matter whether they are correlated or uncorrelated. A preliminary version of this paper will be orally presented in European Test Symposium 2007 as informal digest. However, in this paper, the complete diagnosis flow will be presented for the first time.

The rest of this paper is outlined as follows. Section 2 reviews previous work on non-compressed and compressed pattern diagnosis. Meanwhile we point out the challenges of applying these diagnosis algorithms if compound defects exist on the same die. Section 3 illustrates the proposed compound defects diagnosis methodology. Experimental results on simulated cases and an industrial case study are given in Section 4. The conclusions are drawn in Section 5 followed by some future work discussed in Section 6.

2. Challenges of compound defect diagnosis

2.1 Non-Compressed pattern scan chain diagnosis

Without embedded compressions, a software-based chain diagnosis procedure typically includes the following three steps [GUO01].

Step 1: Identify faulty chains and fault types by chain patterns.

Step 2: Find the upper bound and lower bound of suspect scan cells.

Step 3: Locate the faulty scan cells by injecting fault one cell at a time on the cells within the bounds and simulating the scan patterns.

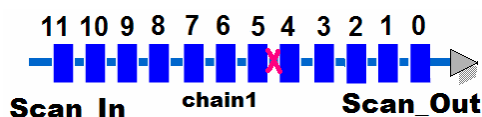


Figure 1: An Example of Scan Chain

The above mentioned algorithm can be illustrated by an example of chain defect diagnosis. As illustrated in Figure 1, suppose a scan chain Chain1 has 12 scan cells numbered from cell 0 to cell 11, where cell 11 is connected to Chain1’s scan input and cell 0 is connected to Chain1’s scan output. Also, suppose there is a stuck-at 1 fault at the scan output pin of cell 5 on Chain1.

In Step 1, we assume Chain1 is loaded with a chain pattern 001100110011, where the leftmost bit is loaded into cell 11 and the rightmost bit is loaded into cell 0. Due to the fault at cell 5, the unloading values are all “1”s. We can infer from the chain test result that there is at least one stuck-at 1 fault on Chain1. But we do not know which scan cell is defective yet.

In Step 2, we will mask the faulty chain by modifying all loading values on Chain1 to “X” during scan pattern simulation. Then we run good machine simulation under the assumption that the system logic is defect-free. Suppose for a scan pattern, we capture “0” in cell 2 and cell 7 on Chain1. Also, we assume cell 7 is a failing bit and cell 2 is a passing bit. That is to say, we observe a “0” at cell 2 and a “1” at cell 7. Therefore we know that the captured “0” in cell 7 had been changed to a “1” during unloading when it passed by the faulty cell location. Hence, we can infer that the faulty cell must be somewhere between the output of cell 7 and input of cell 2. We call Cell 7 the upper bound of the faulty cell and Cell 2 the lower bound of the faulty cell.

In Step 3, fault simulation with all scan patterns is applied to one cell at a time. A fault is injected at each suspect cell within the range calculated in Step2. Loading values in the downstream of this scan cell on this faulty chain (i.e. scan cells between the suspect scan cell and the scan chain output) will be modified

for all scan patterns due to the fault. E.g., a scan pattern has good machine loading value 001110011010 on chain1. If we inject a stuck-at-1 fault on scan cell 3, the loading value will be modified as 00111001111. After applying the capture clock, new values are captured into faulty scan chain chain1. During scan unload process, the captured values in the upstream of the faulty scan cell on this faulty chain will be modified. E.g., if the simulated captured value is 101011101000, the unloading values will be 11111111000. The simulation results will be compared with the observed results from ATE. The best matching cell(s) will be reported as suspect(s).

Note that Steps (2) and (3) are based on the assumption that the system logic is defect free. If this assumption does not hold, the proposed algorithm may not work as to be illustrated next.

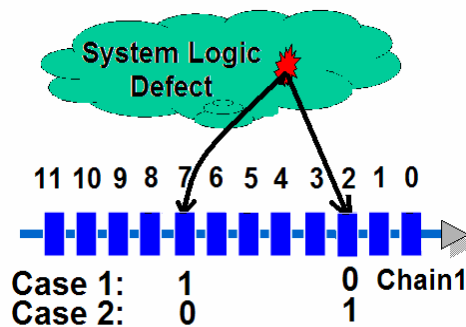


Figure 2: An Example of Compound Defects

We use an example shown in Figure 2 to explain the impact of system logic defect on the above-mentioned scan chain diagnosis algorithm. We still use the same chain defect example used previously, but this time we assume there is also a system logic defect on the same die.

As Case 1 shown in Figure 2, a system logic defect changes the captured values at cell 7 to “1” in the scan pattern we used earlier. In Step 2 of the previously mentioned diagnosis algorithm, we should still observe a failing bit at cell 7 and a passing bit at cell 2 at the Chain1’s scan output. This is OK in this specific example because the system logic defect has no impact on our previous inference that the faulty cell is between cell 2 and cell 7. However, this is not always the case. Let’s consider Case 2 shown in Figure 2, where the system logic defect changes the captured values at cell 2 to “1”. In Step 2 of the previously mentioned diagnosis algorithm, this will lead to an incorrect conclusion that there is a stuck-at 1 fault at the downstream of cell 2.

Even if Step 2 is not impacted, the system logic may change some bits in Step 3. Therefore, if the simulation results may not match with the observation values on ATE, it is very hard to tell whether these mismatches are caused by injecting a

scan chain defect at an incorrect scan cell or by system logic defects.

The above examples illustrate that the previous chain diagnosis algorithm may (as in Case 1) or may not (as in Case 2) work for the failing die with compound defects. We need a more robust diagnosis algorithm to perform the compound defect diagnosis.

2.2 Compressed pattern scan chain diagnosis

Many embedded compression techniques are applied on chips to contain the test cost. Without resorting to bypass mode, compressed pattern diagnosis can be classified into two categories: *indirect diagnosis* and *direct diagnosis* [CHE04]. *Indirect diagnosis* performs diagnosis for compactor-based designs through two phases. In the first phase, the scan cells which should observe failures are identified mathematically from the compactor outputs collected on ATE. In the second phase, with information about which scan cells observe failures from phase 1, any ATPG based diagnostic algorithm that was originally target at circuits without compactors could be applied. *Direct diagnosis* does not need backward mapping to find out failing scan cells. In this paper, we only discuss the *direct diagnosis* algorithms. The same challenges and solutions on compound defects diagnosis are also applicable to *indirect diagnosis*.

A general flow of diagnosing defects in scan chains with compressed patterns includes the following three steps [HUA05].

Step 1: Model a compactor [CHE04] as a function Φ such that $\Phi(R) = r$, where R is the test response before compaction and r is the test response after compaction. The original circuit is then transformed into a circuit with pseudo-scan chains by incorporating compactor function Φ .

Step 2: Read the failure log file for chain test and identify faulty chains and fault types. In this step, masking patterns (explained next) are required to apply in chain test. A masking pattern is a pattern such that only one scan chain is observed from one compactor channel output while all the other scan chains connecting to the same channel output are masked. Same as Step 1 introduced in the previous subsection, it is straightforward to identify faulty chain(s) from pseudo-scan chains by using masking chain patterns.

Step 3: Read the failure log file for scan test. With compressed scan test patterns, apply Step 2 and Step 3 of previously introduced non-compressed pattern chain diagnosis algorithm by performing the fault simulation on faulty scan chain against the transformed circuit. Compare the compacted simulation results with the compacted failure data collected from tester. Report the best matching scan cell(s) as suspect(s).

The above mentioned algorithm can be illustrated by an example of diagnosing a chain defect. Assuming a design with a simple EDT compactor [RAJ02] has two internal scan chains and each chain has 12 scan cells numbered from cell 0 to cell 11. Also assume Chain1 is a good chain while Chain2 is a defective chain with a hold time error between cells 5 and 6.

In Step 1, we transform the circuit with 2 internal chains and a XOR compactor to a circuit with one scan out channel.

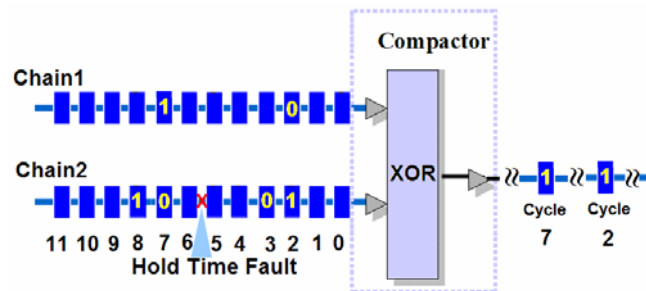


Figure 3: Example of Compressed Pattern Scan Chain Diagnosis without System Defect

In Step 2, masking chain patterns will be used to determine the faulty chain and fault type. Using masking pattern that only observes Chain 2, it is easy to figure out that Chain2 has one hold time fault.

In Step 3, to locate the faulty cell, we utilize system logic scan test patterns. To find the upper bound and lower bound of the faulty cell on Chain2, masking is applied to the faulty chain by changing all loading values on Chain2 to X for a pattern. Then good machine simulation is performed under the assumption that the system logic is defect-free.

Suppose some known values are captured on both Chain1 and Chain2 as shown in Figure 3. Also, we assume at the compactor channel output, cycle 7 is a failing bit and cycle 2 is a passing bit. That is to say, we observe a “1” at cycle 2 and a “0” at cycle 7. Therefore we know that due to the captured transition at cell 7 and cell 8, the captured “0” in cell 7 had been changed to a “1” during unloading when it goes through the hold time fault location. Hence, we can infer that the faulty cell must be somewhere between the output of cell 7 and input of cell 2 on Chain 2. We also call Cell 7 the upper bound of the faulty cell and Cell 2 the lower bound of the faulty cell. After we determine the suspect range on Chain2, we can inject a hold-time fault at the cells in [7, 2], one at a time, and simulate all scan patterns. The simulation results will be compared against the observed results from ATE. The best matching cell(s) will be reported as suspect(s).

Note that Step 3 is based on the assumption that the system logic is defect free. If this assumption is not true, the proposed algorithm may not work as to

be illustrated next. We use one example shown in Figure 4 to explain the impact of system logic defect on the previous published compressed pattern scan chain diagnosis algorithms.

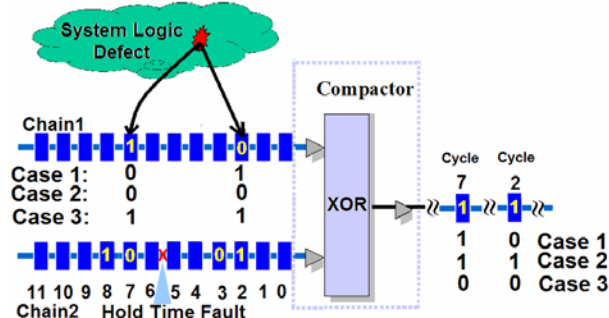


Figure 4: Example of Compressed Pattern Scan Chain Diagnosis with System Defect

Assume there is a system logic defect on the same die, the prior compressed pattern chain diagnosis algorithm [HUA05] may lead to wrong diagnosis. For the example shown in Figure 4, the system defect may have different impact on chain diagnosis in different situations. We only use the following three situations as examples:

(1) In Case 1, the system logic defect changes the captured values at cell 2 and cell 7 on Chain1. Cells (7, 2) on Chain1 will unload (0, 1) and Cells (7, 2) on Chain2 will unload (1, 1) due to the hold-time error. After compaction, we will observe Cycles (7, 2) is (1, 0) at the compactor channel output. That is to say, we will see a failing bit at cycle 2 and a passing bit at cycle 7 at the compactor output channel. This would lead to an incorrect conclusion that there is an intermittent hold time [HUA03] fault at the downstream of cell 2 at Chain2. The fault is intermittent because it only makes the Cell 2 on Chain 2 failed during unloading, but it did not fail the Cell 7 on Chain 2 during unloading.

(2) In Case 2, the system logic defect only changes the captured values at cell 7 on Chain1. Cells (7, 2) on Chain1 will unload (0, 0) and Cells (7, 2) on Chain2 will unload (1, 1) due to the hold-time error. After compaction, we will observe Cycles (7, 2) is (1, 1) at the compactor channel output. That is to say, we will see both Cycle 2 and Cycle 7 are passing bits at the compactor output channel. This would lead to an incorrect conclusion that the hold-time fault is in the upper stream of Cell 7 on Chain2.

(3) In Case 3, the system logic defect only changes the captured values at cell 2 on Chain1. Cells (7, 2) on Chain1 will unload (1, 1) and Cells (7, 2) on Chain2 will unload (1, 1) due to the hold-time error. After compaction, we will observe Cycles (7, 2) is (0, 0) at the compactor channel output. That is to say, we will see both Cycle 2 and Cycle 7 are failing bits at the compactor output channel. This would lead to an

incorrect conclusion that the hold-time fault is in the down stream of Cell 2 on Chain2.

The system logic defects may also change the captured values at other cells such as Cells 2, 3, 7 8 on Chain2 or their combinations. The impact of system logic defect could invalidate the previously published compressed pattern chain diagnosis in numerous ways.

2.3 System logic diagnosis

In all previous published work on scan based system logic diagnosis, it is assumed that the scan chains are defect-free. When the assumption is not valid in reality, the previous logic diagnosis methods will not work.

In [HUA01] [WAN06], multiple fault diagnosis was investigated. In [HUA01] heuristics for grading each signal's defect possibility were proposed by using curable outputs and curable vectors. They proposed a grading criterion combining these two measures to produce a more accurate estimation for each signal's defect possibility. The proposed heuristics can be used when the total number of faults in a failing chip is very small.

In [WAN06] failures are partitioned into groups such that the failures in each group are affected by only one or a few faults. The partition usually can be preformed successfully based on their observation that different faults usually have disjoint failing points. This is true when the number of total faults is very small.

With defective scan chain(s), one pattern could have a large number of faulty loading / unloading points which can be treated as if it has multiple "logic faults". However, since the number of such "logic faults" could be very large due to scan chain fault(s), the previous published multiple fault diagnosis and failure partitions will not work in practice because a large number of faults may interact with each other.

In the next section, we will discuss how to diagnose the compound defects by proposing an innovative diagnosis flow.

3. Compound defects diagnosis

3.1 Failure partition

As we already know, we can identify faulty chain based on chain test patterns. However, it is not straightforward to identify if a failing chip has both scan chain defects and system logic defects. To achieve the goal, we try to partition all the failure bits into two groups. One group (called Group1) of failure bits must be caused by system logic failures and the other group (called Group2) of failure bits could be caused by either chain defects or logic defects or their compound effects.

Once the partition is successful, we can run logic diagnosis on failure bits in Group1 while masking the faulty chain(s) and run chain diagnosis on failures in Group2 while masking the system logic defect(s). If failure bits partition cannot be done, it will do chain diagnosis only. To improve resolution of this situation, it may have to run per-pattern based chain diagnosis, which will be explained in subsection 3.3.

To partition the failure bits, we propose a new algorithm in four steps.

(1) Identify all faulty chains and good chains by chain test patterns. Assuming if a chain never failed chain patterns, it is a good chain. Otherwise, it is a faulty chain. If embedded compression technique is used, identify all faulty channels and good channels. A faulty channel is connected through the compactor with at least one faulty chain. Otherwise, it is called a good channel.

(2) We modify the design netlist by changing all scan cells on all faulty chains to X-mask gates. The X-mask gate has the property that no matter what the input of the gate is, it always loads X at the input and produces X at the output of the gate for each time frame.

(3) Simulate all scan patterns against the modified design netlist under the assumption that no system logic defects.

(4) Identify if there are any failure bits induced by the system logic defects. If we found any such failure bits, we put them into Group1 and put the rest failure bits into Group2. A failure bit is called system logic defect induced if it satisfies the following two conditions simultaneously:

Condition A: The failure bit is on either a good scan chain (or a good scan channel in case embedded compression technique is used) or at a PO.

Condition B: The simulation result at this failing bit in step (3) is not X.

If *condition A* is satisfied, we know that the failing bit is not caused during unloading procedure because it is on good chains / channels or POs. Meanwhile, if *condition B* is satisfied, we infer that the failing bit is not caused by faulty scan chain load values because no X was propagated to this bit in scan pattern simulation, which means any faulty values on any faulty chains have no impact on the value captured at this bit. Therefore, the failure bit must be caused by system logic defect. At this point, we know that the failing chip has compound defects and the failure partition is successful.

Note that it is possible that failure partition is unsuccessful for a failing chip with compound defects because all system defect induced failure bits are masked by scan chain failures. For example, if a system defect only produces failure bits on faulty chains, we cannot identify it has compound defects. As another example, a system defect produces some

failing bits on a good chain. However, the incorrect loading values on the faulty chain also make these bits failed on this good chain. In that scenario, we cannot make sure what is exactly the cause of these failing bits. In case we cannot partition the failure bits into two groups, it has two scenarios: (1) the system logic defects have no impact on chain diagnosis results such that chain diagnosis still get correct result and (2) the system logic defects do have impact on chain diagnosis results such that chain diagnosis get incorrect diagnosis result. For scenario (2), we still have a chance to enhance the chain diagnosis result by using per-pattern based diagnosis as explained later.

3.2. System logic diagnosis with defective chains

Suppose we can successfully identify all failures that induced by system logic defects, we run logic diagnosis first by using a novel diagnosis flow as shown in Figure 5.

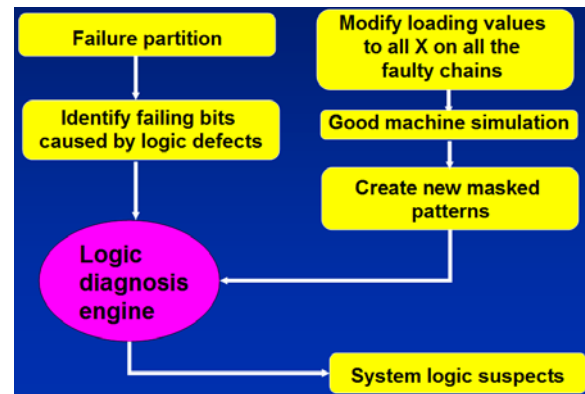


Figure 5: System Logic Diagnosis with Defective Scan Chains

In the proposed diagnosis flow we use a conventional logic diagnosis engine (shown in the oval box). We apply two sets of input information to this engine. The first set of information is the failure bits caused by the system logic defects as we explained in the previous subsection. The second set of information is the modified patterns by taking the defective scan chain into consideration. To modify scan patterns for logic diagnosis in the context of faulty chains, we modify all scan loading values to X on all faulty chains. We then run good machine simulation with all the modified patterns against the modified design netlist to create new masked patterns. In the new pattern set, all loading / unloading values on the whole faulty chain(s) are masked to X. This masking is called *full-chain masking*, which makes logic diagnosis results independent of scan chain diagnosis. When the logic diagnosis is done we change all X-mask gates back to normal scan cells.

Traditionally, ATPG and fault simulation are under the assumption that scan chains are perfectly functioning. If this assumption does not hold, the generated patterns cannot be reliably applied to the chip. Usually, a re-spin is necessary to re-design scan chains and produce chips with perfectly functioning chains. However, re-spin is expensive. In the above proposed diagnosis algorithm, the system logic diagnosis can be applied as long as the faulty chain(s) is masked. If we identify some system logic issues caused by design or manufacturing process, the next re-spin can fix both chain defect and system logic defect at one shot.

3.3 Chain diagnosis with defective system logic

There are three scenarios for chain diagnosis in the context of diagnosing compound defects.

Scenario 1:

Both the failure partition and logic diagnosis are successful. Here, we heuristically define the success of logic diagnosis if the logic diagnosis (1) has located at least one logic suspect, and (2) at least one suspect has score 100, i.e., the suspect can perfectly explain all failures in Group 1. In this case, we mask all system logic gates in the logic diagnosis suspect report by changing these gates to X-mask gates. We then run previously proposed chain diagnosis on the failures in Group 2 against the modified design.

Scenario 2:

The failure partition is successful, but the logic diagnosis is unsuccessful. Here, we heuristically define the fail of logic diagnosis if the logic diagnosis (1) cannot locate any logic suspect, or (2) no suspect has score 100. In this case we have to mask all system logic that may potentially contribute to the failure bits in Group1. It applies three steps:

(1) We use the critical path tracing algorithm described in [AKE90] to find all the initial candidates in system logic by back tracing from each failure bit in Group 1. For a failure bit k , we get a candidate logic signal set S_k .

(2) We get a logic signal set S , which includes all potential defective logic signals. We use either union or intersection operations: $S = \cup_k S_k$ or $S = \cap_k S_k$.

(3) We mask all system logic gates in set S , by changing these gates to X-mask gates.

In (2), performing union operation will guarantee all the real system defects are included in S , even it has multiple system logic defects. The downside of this method is that it may mask too many system logic gates, which may result in dropped chain diagnosis resolution.

As the example illustrated in Figure 6, perform union operation after tracing back from failures Z_1 ,

Z_2 , Z_3 in Group1 will guarantee the real system defects are included in S , even it has multiple system logic defects.

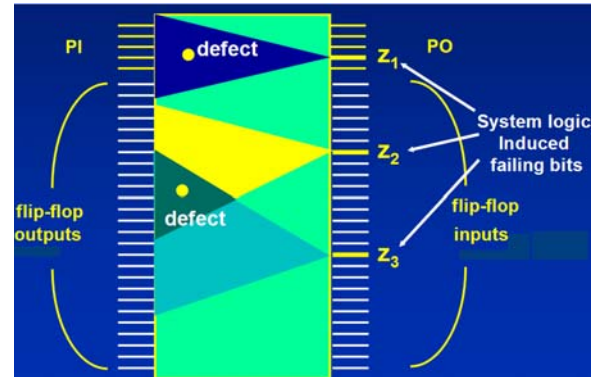


Figure 6: Use Union to Find System Logic Defects Candidates

Performing intersection operation will not guarantee the real system defects are all included in S , unless it has single system logic defect. The advantage of this method is it has better chain diagnosis resolution because lesser system logic gates are masked.

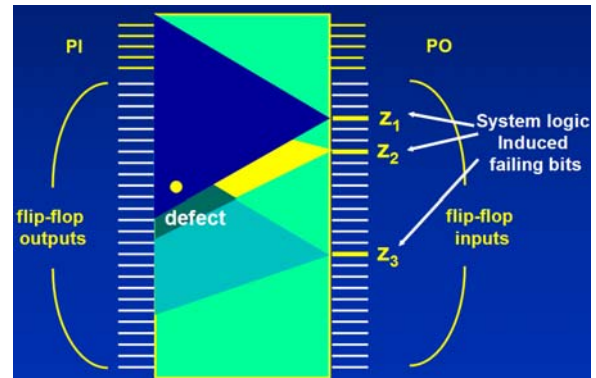


Figure 7: Use Intersection to Find System Logic Defects Candidates

As the example illustrated in Figure 7, perform intersection operation after tracing back from failures Z_1 , Z_2 , Z_3 in Group1 will have the real system defect included in S only if it has single system logic defect.

In reality, since we don't know whether it has single or multiple system logic defects, we have two options when identifying the set of logic signals to be masked.

(1) Use union operation first. If the union operation results in unacceptable scan chain diagnosis resolution, we can try intersection operation.

(2) Use intersection operation first. If the intersection is empty, it means it must have multiple system defects. We can change to union operation. If the intersection is not empty, we assume it has single

system logic defect at the moment. Note that it is still possible it has multiple system logic defects. If this is true, the scan chain diagnosis will not find any chain suspects with perfect match. When that happens, we may revert to union operation.

Scenario 3:

When the failure bits partition is unsuccessful, we run chain diagnosis under the assumption that system logic is defect free. As we pointed out earlier, it is possible that all system logic defect induced failures are masked by scan chain failures. The masking only happened when we change the scan cells on faulty chains to X-mask gates. In chain diagnosis, the X-mask gates are changed back so the system logic defects may have impact on the diagnosis results. We rely on chain diagnosis suspect scores to identify the impact of system logic defects. When we calculate a suspect score, we inject the fault at the suspect location and run simulation. If all simulated values are perfectly matched with the observed values, we give the suspect score 100. Note that if we use notation of “simulated_value <-> observed_value”, only “0<->1” and “1<->0” are considered mismatch, “X<->1” and “X<->0” are not considered mismatch. If the diagnosis cannot find any suspect with score 100, it means there must be at least one system logic defect co-exist on the chip. Since we cannot partition the failure bits, we have to use per-pattern based diagnosis proposed next.

The per-pattern based diagnosis algorithm is based on the observation that system logic defect may only impact a subset of scan patterns while chain defect could impact all scan patterns. Therefore for each failing pattern, chain diagnosis would find a set of suspect cells associated with some scores. If the system logic defect impacts one pattern, the scores associated with the suspects identified by this pattern must be less than 100. If the score of a suspects identified by a pattern is 100, it means it did not trigger the system logic defects for this specific pattern. Hence, the scores can be used as a filter to screen out possible incorrect suspects caused by the system logic defects.

In the next subsection, we will use a diagram to illustrate the entire diagnosis flow by putting together every technique proposed earlier.

3.4 Compound defects diagnosis flow

The entire proposed compound defects diagnosis flow is illustrated in Figure 8. The flow starts from the failure partition. If failure partition is successful, we run logic diagnosis. If logic diagnosis is successful (Scenario 1), we run chain diagnosis after masking logic suspects. If logic diagnosis is unsuccessful (Scenario 2), we run chain diagnosis after masking all gates in the fan-in cones of failures

in Group1 by taking union / intersection operations. If failure partition is unsuccessful (Scenario 3), we may have to rely on per-pattern based diagnosis.

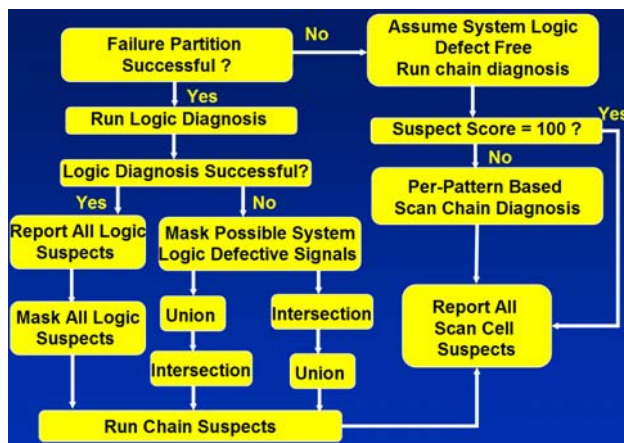


Figure 8: Compound Defects Diagnosis Flow

4. Experimental results

4.1 Simulated Cases

This section presents experimental results for the proposed compound defects diagnosis by using simulated cases.

First we use a non-compressed industrial design with 8 scan chains. The scan chain lengths vary from 655 to 725. We run conventional ATPG targeting stuck-at fault to generate one chain pattern and 500 scan patterns. We create 1000 simulated cases with compound defects. For each case, we randomly pick a scan cell and inject a stuck-at-0 chain fault at this cell. Meanwhile, we randomly pick a gate from system logic and inject a stuck-at-0 fault at the output of this gate. After injecting a fault pair for each case, we run simulation of one chain pattern and 500 scan patterns to generate a simulated fail log. We then run two diagnosis algorithms on each case and compare the diagnosis results. The first chain diagnosis algorithm (Alg1) is based on an algorithm similar to [GUO01] [HUA03], where the system logic is assumed defect-free. The second algorithm (Alg2) is the one proposed in this paper.

If the reported suspects include the injected fault site, it is called a hit. The *Chain_Diag_Hit_Rate* and *Logic_Diag_Hit_Rate* are defined as follows.

$$Chain_Diag_Hit_Rate = \frac{\# \text{ of hitted chain diagnosis}}{\# \text{ of Total cases}}$$

$$Logic_Diag_Hit_Rate = \frac{\# \text{ of hitted logic diagnosis}}{\# \text{ of Total cases}}$$

As indicated in Table1, using the proposed algorithm the *Chain_Diag_Hit_Rate* is increased from 37.1% to 90.4%, and the *Logic_Diag_Hit_Rate* is increased from 0 to 58.4%. That is to say, about 58% of the cases, we can successfully diagnose both chain and system logic defects, 32% cases, we can successfully diagnose chain defects, but we cannot get any suspects for the logic defects. There are also less than 10% cases, we either cannot get any suspects or we get misdiagnosed results.

Table 1: Diagnosis Hit Rate for a Non-Compressed Design

	Alg1[GOU01]	Alg2
<i>Chain_Diag_Hit_Rate</i>	37.1%	90.4%
<i>Logic_Diag_Hit_Rate</i>	0	58.4%

The unsuccessful diagnosis of logic defect is because the unsuccessful failure partition. That is to say, there are about 42% cases such that when we mask the entire faulty chain to X-mask gates, all failures bits induced by the logic defects got X. Therefore the failure partition algorithm proposed in Section 3.1 can only be performed on about 58% of these specific cases. The failure partition success rate is determined by the circuit, scan chain architecture and patterns. Since the circuit we used has only 8 scan chains, masking one chain may mask out a large number of logic cones. It is expected that when the number of scan chains is increased, the masking effects induced by one chain will be reduced.

The unsuccessful diagnosis of chain defect is because when the failure partition is unsuccessful, some failure bits caused by the logic defect happened to mislead the chain diagnosis to consider the faulty chain has multiple or intermittent chain fault. If it happens to find perfect match with an incorrect chain fault model, it will not run per-pattern based diagnosis. To enhance the diagnosis hit rate we may force the chain diagnosis run per-pattern based diagnosis even it already found a suspect with score 100.

As we know, when failure partition is unsuccessful, we cannot diagnose logic defects. However, we may still perform chain diagnosis successfully. We further analyze the cases where logic defects are masked, but chain diagnosis is successful. These cases can be classified into two groups. In the first group (denoted as G1), the injected logic defect has no impact on chain diagnosis. An example of such case is shown in Case 1 of Figure 2. In the second group (denoted as G2), the injected logic defects do have impacts on chain diagnosis. However with the proposed per-pattern based diagnosis algorithm we can filter out the impact and still get correct chain diagnosis results.

As shown in Table 2, when failure bit partition failed, we have 320 case can be diagnosed correctly for chain defects. Among them, almost 95% cases the logic induced defects have no impact on chain diagnosis. Only about 5% cases that we have to go through per-pattern based diagnosis to screen out the impact of system logic. This is because the number of failure bits caused by chain failure defects is much larger than the number of failure bits caused by system logic defects such that in most cases system logic induced failures are masked by scan chain failures when failure bits partition failed. Although per-pattern based diagnosis is only applied in a small number of cases, it is effective to correctly diagnose those difficult cases where failure bits partition is impossible, but system logic defects disturb the conventional chain diagnosis flow.

Table 2: Separate G1 and G2 for a Non-Compressed Design

# of cases with unsuccessful logic diagnosis, but successful chain diagnosis	G1 (logic defect has no impact)	G2 (per-pattern based chain diagnosis)
320	94.7%	5.3%

Next we use an industrial design with EDT technology [RAJ02] to run some experiments. The design we used has 128 internal scan chains and 8 external compactor channels. So the compaction ratio is 16X. The internal scan chain lengths vary from 458 to 508. We run conventional ATPG targeting stuck-at fault to generate a few chain patterns and 500 scan patterns. We create 1000 simulated cases with compound defects. For each case, we randomly pick an internal scan chain and a scan cell to inject a stuck-at-0 chain fault at this cell. Meanwhile, we randomly pick a gate from system logic and inject a stuck-at-0 fault at the output of this gate. After injecting a fault pair for each case, we run simulation of all chain patterns and the first 32 scan patterns to generate a simulated fail log. We then run two diagnosis algorithms on each case and compare the diagnosis results. The first chain diagnosis algorithm (Alg1) is based on an algorithm proposed in [HUA05], where the system logic is assumed defect-free. The second algorithm (Alg2) is the one proposed in this paper. We measure *Chain_Diag_Hit_Rate* and *Logic_Diag_Hit_Rate* defined earlier.

As indicated in Table3, using the proposed algorithm the *Chain_Diag_Hit_Rate* is increased from 33.6% to 85.9%, and the *Logic_Diag_Hit_Rate* is increased from 0 to 56.8%. That is to say, about 57% of the cases, we can successfully diagnose both chain and system logic defects, about 29% cases, we can successfully diagnose chain defects, but we

cannot get any suspects for the logic defects. There are also about 14% cases, we either cannot get any suspects or we get misdiagnosed results.

Table 3: Diagnosis Hit Rate for an EDT Design

	Alg1[HUA05]	Alg2
<i>Chain_Diag_Hit_Rate</i>	33.6%	85.9%
<i>Logic_Diag_Hit_Rate</i>	0	56.8%

The reasons for unsuccessful diagnosis of non-compressed designs we analyzed earlier are also applicable to compressed designs. We also find that masking one internal scan chain will mask all scan chains that use the same compactor channel. This makes the success rate of failure partition dropped. However, even with the impact of embedded compactors, we can see the diagnosis hit rate has dramatic increase by using the proposed compound defects diagnosis flow.

4.2 An industrial case study

We successfully applied the proposed compound defect diagnosis method to a few failed chips of an industrial design. PFA validated that the diagnosis results are correct on these failed chips. The chips were designed by AMD and manufactured with 90nm TSMC process. The design has about 10M gates and about 1M scan cells. EDT technology has been adopted for the manufacturing test of the chip.

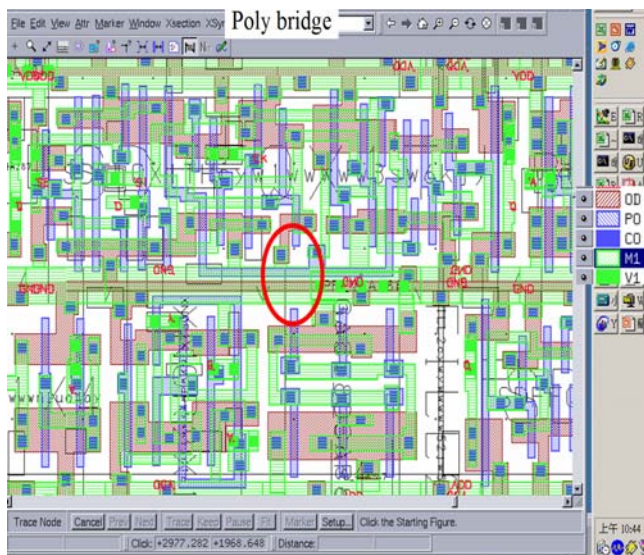


Figure 9: Layout of an example of a real case

Figure 9 illustrates the layout of the defective location of one failing chip, where one end of the poly bridge is at scan chain and the other end of the bridge is at the system logic. Pattern simulation showed that both chain defect and system logic

defect were triggered during test. With the proposed diagnosis flow, we partitioned the failures successfully. Scan chain diagnosis and logic diagnosis are successful for this test case. Physical FA confirmed diagnosis results on silicon. Without this compound diagnosis flow, only scan chain diagnosis would have been performed on this test case and diagnosis results would be misleading.

5. Conclusions

In this paper, we analyzed the challenges for diagnosing circuits with compound scan chain defect and system logic defect. We then proposed a new diagnosis algorithm and a new diagnosis flow that can be used to diagnose the compound defects. We partition the failures first. We mask the faulty scan chain(s) to diagnose system logic defects and mask the system logic defects to diagnose scan chain(s) defects. Experimental results with simulated cases have demonstrated the effectiveness of the proposed techniques. The proposed method was validated with a real case of diagnosing compound defects.

6. Future work

When the failure partition is successful, we actually have three options in the diagnosis flow:

- (1) Each diagnosis does not depend on the result of the other diagnosis.
- (2) Run logic diagnosis first and use the logic diagnosis results to enhance chain diagnosis.
- (3) Run chain diagnosis first and use the chain diagnosis results to enhance logic diagnosis.

In this paper we proposed to apply flow (1) if logic diagnosis is unsuccessful. We also proposed to use flow (2) when logic diagnosis is successful. In fact, we also did some extra experiments for flow (3). However, the results are not as good as the results in flow (2) due to masking too many logic gates.

In future, iteration between flows (2) and (3) should be investigated as well. An interactive system logic diagnosis and scan chains diagnosis flow can be used to enhance diagnosis resolution. An example of such idea is shown in Figure 10, in which we start logic diagnosis first after masking the entire faulty chain. We then run chain diagnosis after masking the gates in the logic diagnosis report as we proposed in this paper. Suppose after chain diagnosis, we identify the defect location of the faulty chain. We may only mask the loading values on the scan cells from the fault location all the way down to the scan cell connected with the scan output. Meanwhile, we may only mask the unloading values on the scan cells from the fault location all the way up to the scan cell connected with the scan input. The advantage of this method is that *partial-chain masking* may achieve better logic diagnosis resolution than the previously

proposed *full-chain masking* method. The loop is stopped when the diagnosis resolution enhancement is smaller than a predefined limit or the user defined run time limit is reached.

Similarly another example of such idea is shown in Figure 11, in which we start chain diagnosis first. We then run logic diagnosis with partial-chain masking and use the logic diagnosis result to enhance chain diagnosis results. The loop is stopped when the diagnosis resolution enhancement is smaller than a predefined limit or the user defined run time limit is reached.

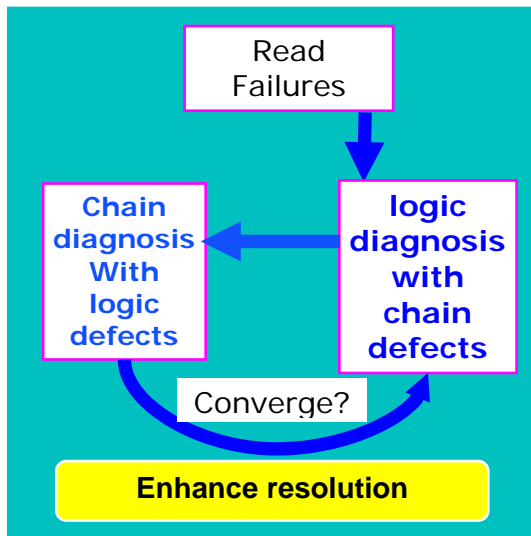


Figure 10: Interactive Logic Diagnosis + Chain Diagnosis

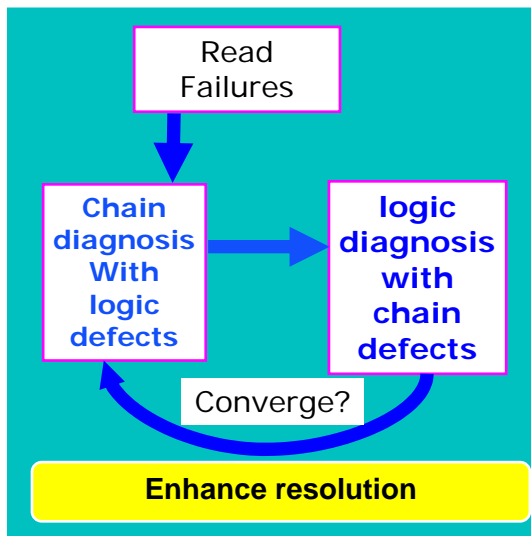


Figure 11: Interactive Chain Diagnosis + Logic Diagnosis

REFERENCES

- [AKE90] S.B. Akers, B. Krishnamurthy, S. Park and A. Swaminathan, "Why Is Less Information From Logic Simulation More Useful in Fault Simulation," Proc. Int'l Test Conference, 1990, pp. 786-800.
- [CHE04] W.-T. Cheng, K.-H. Tsai, Y. Huang, N. Tamarapalli and J. Rajski, "Compressor Independent Direct Diagnosis," Proc. Asian Test Symp. 2004, pp. 204-209.
- [CRO05] A. Crouch, "Debugging and Diagnosing Scan Chains," EDFAS, Vol. 7, Feb., 2005, pp 16-24.
- [EDI95] S. Edirisooriya and G. Edirisooriya, "Diagnosis of Scan Failures," Proc. VLSI Test Symposium 1995, pp. 250-255.
- [GUO01] R. Guo and S. Venkataraman, "A Technique for Fault Diagnosis of Defects in Scan Chains," Proc. Int'l Test Conference, 2001, pp. 268-277.
- [HUA01] S.-Y. Huang, "On improving the Accuracy of Multiple Defect Diagnosis," Proc. VLSI Test Symposium 2001, pp. 34-39.
- [HUA03] Y. Huang, W.-T. Cheng, S.M. Reddy, C.-J. Hsieh, Y.-T. Hung, "Statistical Diagnosis for Intermittent Scan Chain Hold-Time Fault," Int'l Test Conference, 2003, pp.319-328.
- [HUA04] Y. Huang, W.-T. Cheng, K.-H. Tsai, G.Crowell and C. McMahon, "Diagnosing DACS (Defects That Affect Scan Chain and System Logic)," ISTFA, 2004.
- [HUA05] Y. Huang, W.-T. Cheng, and J. Rajski, "Compressed Pattern Diagnosis For Scan Chain Failures," ITC 2005.
- [KUN94] S. Kundu, "Diagnosing Scan Chain Faults," IEEE Trans. On VLSI Systems, Vol. 2, No. 4, December 1994, pp. 512-516.
- [NAR97] S. Narayanan and A. Das, "An Efficient Scheme to Diagnose Scan Chains," Proc. Int'l Test Conference, 1997, pp. 704-713.
- [RAJ02] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee, R. Thompson, H. Tsai, A. Hertwig, N. Tamarapalli, G. Eide, and J. Qian, "Embedded Deterministic Test for Low Cost Manufacturing Test," Proc. ITC, pp.301-310, 2002.
- [SCH92] J. Schafer, F. Policastri and R. McNulty, "Partner SRLs for Improved Shift Register Diagnostics," Proc. VLSI Test Symposium, 1992, pp.198-201.
- [WAN06] Z. Wang, M. Marek-Sadowska, K.-H. Tsai, J. Rajski, "Analysis and methodology for multiple-fault diagnosis," IEEE Trans. on CAD, Volume 25, Issue 3, March 2006 pp 558 - 575.
- [WU98] Y. Wu, "Diagnosis of Scan Chain Failures," Proc. Int'l Symp. on Defect and Fault Tolerance in VLSI Systems, 1998, pp.217-222.
- [YAN05] J.-S. Yang and S.-Y. Huang, "Quick Scan Chain Diagnosis Using Signal Profiling", Proc. of Int'l Conf. On Computer Design, Oct., 2005, pp. 157-160.