Intermittent Scan Chain Fault Diagnosis Based on Signal Probability Analysis

Yu Huang[1], Wu-Tung Cheng[1], Cheng-Ju Hsieh[2],
Huan-Yung Tseng[2], Alou Huang[2], Yu-Ting Hung[2]
[1] Mentor Graphics Corporation, 8005 S.W. Boeckman Rd., Wilsonville, OR 97070
{Yu_Huang, Wu-Tung_Cheng}@mentorg.com
[2] Diagnosis Technique & Design Development Division, Faraday Technology Corporation
{cjhsieh, huanyung, alouh, adrian}@faraday.com.tw

Abstract

A new algorithm to diagnose intermittent scan chain fault in scan-based designs is proposed in this paper. An intermittent scan chain fault sometimes is triggered and sometimes is not triggered during scan chain shifting, which makes it very difficult to locate the fault sites. In this paper, we provide answers to three questions:

(1) Why intermittent scan chain faults happen?
(2) Why diagnosis of this type of faults is necessary?
(3) How to diagnose this type of faults?

The experimental results presented demonstrate that the proposed diagnosis algorithm is effective for large industrial designs with multiple intermittent scan chain faults.

1. Introduction

Scan-based design has become the most pervasive DFT technique used in the VLSI industry today. Scan chains that function correctly are the first requirement to test and diagnose the complete circuit. Therefore diagnosing faulty scan chains is a critical step to reduce the time of silicon debug for new products and improve yield for mass production.

Previous works [1]-[5] in scan chain diagnosis only target at permanent scan chain faults. It was assumed that if a fault happens at a scan cell, it is permanently triggered during chain shifting. However for the modern industrial VLSI at UDSM level, people are sometimes confronted with another type of scan chain fault — intermittent scan chain fault that are triggered intermittently. Its intermittent behavior makes the diagnosis of the fault locations extremely difficult. Recent research [6][7] investigated this problem. In [6], an upper bound and a lower bound of the faulty cell locations are identified in the first step, followed by a statistical method to rank each candidate faulty cell location in the second step. However, the method proposed in [6] cannot handle the diagnosis problem if multiple intermittent faults are in the same scan chain. In [7] an algorithm based on X-fault simulation is proposed to diagnose multiple intermittent faults on one scan chain. In this paper, we propose a different approach to solve the same problem as in [7]. In addition, unlike in [6] and [7], where only intermittent scan chain hold-time faults are investigated, several types of intermittent scan chain faults are considered in this work.

The paper is organized in the following manner. In Section 2 we analyze various possible root causes for the intermittent scan chain faults and explain the necessity of intermittent scan chain fault diagnosis. In Section 3 a diagnosis algorithm based on signal probability analysis is proposed. Experimental results on industrial VLSI designs are presented in Section 4, followed by the conclusions in Section 5.

2. Intermittent Scan Chain Faults

2.1. Scan Chain Fault Types

The type of a defective scan chain is typically identified by a few chain flush patterns. As an example given in Table 1, suppose a scan chain is shifted in with a chain flush pattern 001100110011, Column 2 gives the unloaded faulty values for each type of permanent fault. Column 3 gives examples of the unloaded faulty values for each type of intermittent fault. Note that the Xs depend on the previous or the next chain flush pattern.

Table 1: Various Scan Chain Fault Types

Fault Types	Unloaded Values-Permanent Faults	Unloaded Values-Intermittent Faults
Stuck-at-0	000000000000	001000010000
Stuck-at-1	111111111111	101111111011
Slow-to-Rise	00100010001X	00110010001X
Slow-to-Fall	01110111011X	01110011011X
Slow	01100110011X	00100111011X
Fast-to-Rise	X01110111011	X01110110011
Fast-to-Fall	X00100010001	X00100110001
Fast	X00110011001	X00100111001

The *permanent* stuck-at faults usually appear when the scan chain is broken or it is shorted with ground or power lines. The *intermittent* stuck-at faults are often caused by short between a signal on scan path and a signal on system path. The timing-related faults in Table 1 could occur when two adjacent scan cells that are shifted in opposite values. Slow faults are caused by slow transitions during scan shifting. They are usually called *Transition Faults*. Fast faults are typically caused by hold-time violations. Thus they usually called *Hold-Time Faults*. There are various causes for intermittent timing related faults, which are discussed in the next sub-section.

2.2. Root Causes of Intermittent Scan Chain Transition Faults and Hold-Time Faults

(1) Process Variation. Process integration engineers are gradually failing to keep process variations hidden behind the defensive barrier of tight design rules. Variations in metal line widths, layer thicknesses, via integrity and transistor critical dimensions, while continually shrinking in absolute terms, are growing as a percentage of the increasingly minuscule dimensions they affect. The mechanisms that lead to these variations seem to be growing in number as processes become more complex [8]. Therefore gate delay and wiring delay in a single die will become difficult to calculate accurately by EDA tools and difficult to control in process.

(2) Limitations of the current EDA tools. At UDSM levels of technologies, wiring delay accounts for the vast majority of overall delay. It is known that by 90 nm, wiring delay will account for about 75% of the overall delay [9]. Therefore at 90 nm and below, performance of a VLSI is unknown prior to detailed routing, which makes the traditional timing analysis / verification inaccurate. It is quite difficult, if not impossible, that the delay calculated by EDA tools exactly matches the delay in real silicon. This problem is further exacerbated by signal integrity (SI) and design integrity (DI) issues (to be explained below) that cannot be ignored in the modern VLSI production.

(3) Design Libraries. Even for the same VLSI process, different libraries would provide memory elements with different hold-time margins and library cells with different timing variation margins.

(4) Crosstalk, which is one of the Signal Integrity (SI) issues. To increase density, interconnects on the chip come closer and are made narrower and thicker. Hence the crosstalk is exacerbated by the increased inter-line capacitive and inductive coupling. In this scenario, the gate / wiring delay becomes more susceptible to crosstalk. Crosstalk might speedup a signal transition if the coupled lines have the same transitions. Under this situation the *Hold-Time Faults* might intermittently be triggered because of intermittent hold-time violations caused by fast transitions. This is especially true when *hold-time margin* becomes very small due to process variation. Similarly, crosstalk might slow down a signal transition if the coupled lines have the opposite transitions. Under this situation the *Transition Faults* might intermittently be triggered because of intermittent slow transitions.

(5) IR drop, which is another SI issue. Increased switching activity during the test pattern application might cause lowering supply voltages, which is called IR drop. Based on our experiences, even 10% of IR drop will vary the wire delay, which makes the calculated timing inaccurate. If the *timing variation margins* (e.g., hold-time margin) are not big enough to tolerate such variations, *Hold-Time* or *Transition Faults* may "intermittently" be triggered at some IR drop areas.

(6) Wire self-heating, which is one of the Design Integrity (DI) issues. The temperature variations, which might be the result of wire self-heating, could change wiring timing as well.

What we listed above are the major causes that are possibly responsible for the intermittent scan chain timing-related faults. There might be some other causes not discussed here.

2.3. Intermittent Fault Diagnosis is Necessary.

Two reasons make diagnosis of the intermittent scan chain fault necessary.

(1) The problem appears much more frequently than before with the development and application of UDSM technologies in recent years. A few years ago, ignoring the impacts of process variations, crosstalk, IR-drop etc. would only incur small yield loss. The cost overhead for analyzing these causes was too big to be justified by the yield improvement. However, we cannot afford to ignore these problems any more.

(2) Global fixing of the problem without knowing the fault sites has drawbacks. The traditional workarounds include:

(i) Globally raise the power rail.
(ii) Globally raise the *hold-time margin*
(iii) Globally slow down the clock speed.

Method (i) may avoid the intermittent scan chain faults introduced by IR drop. However it cannot avoid intermittent fault that caused by other problems. Moreover, raising power rail globally would introduce unnecessary overhead.

To fix even a single *Hold-Time Fault*, method (ii) may unnecessarily add buffers to increase delays in the scan paths all over the chip, which leads to increased silicon area overhead. In addition, inappropriate delay insertion would introduce setup-time violations.

To fix even a single *Transition Fault*, method (iii) may unnecessarily lower the clock speed during testing, which would lead to longer test time and higher test cost.

Due to the drawbacks of the above-mentioned methods, intermittent scan chain fault diagnosis is necessary, which allows us (i) to fix the problem locally with low cost, (ii) to possibly identify the root causes of the problem, and (iii) to prevent the fault from appearing in the next release of the design or next volume of production. A diagnosis algorithm based on signal probability analysis is introduced next.

3. Proposed Diagnosis Algorithm

To simplify the illustration of the algorithm, we make the following assumptions.

(1) Each faulty scan chain could have multiple intermittent scan chain faults of the same type. Multiple faulty scan chains could exist in a design. Different scan chain may have different fault types.

(2) The system logic out of scan chains is fault-free.

(3) Same as in previous work [1][2][6][7], chain flush patterns are used to identify faulty chains and fault types. Hence it is not discussed in this paper.

To compare the proposed algorithm with the algorithm proposed previously in [7], a brief review of the old algorithm will be presented in Subsection 3.1. Then we will illustrate how to diagnose single faulty chain with single fault by the proposed algorithm in Subsection 3.2. This is followed by the extensions of the method for multiple faults on single chain and multiple faulty chains in Subsection 3.3 and 3.4 respectively.

3.1. Review of the Algorithm Proposed in [7]

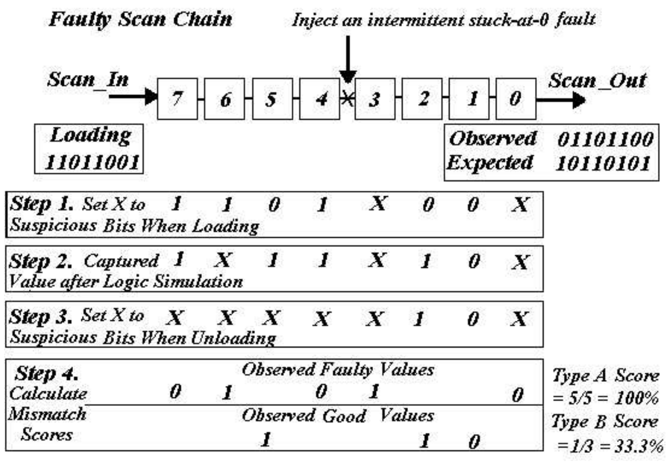

Faulty Scan Chain
Inject an intermittent stuck-at-0 fault
Scan_In
7 6 5 4 3 2 1 0
Scan_Out
Loading 11011001
Observed 01101100
Expected 10110101
Step 1. Set X to Suspicious Bits When Loading 1 1 0 1 X 0 0 X
Step 2. Captured Value after Logic Simulation 1 X 1 1 X 1 0 X
Step 3. Set X to Suspicious Bits When Unloading X X X X X 1 0 X
Step 4. Calculate Mismatch Scores
Observed Faulty Values 0 1 0 1 0
Observed Good Values 1 1 0
Type A Score = 5/5 = 100%
Type B Score =1/3 = 33.3%

Figure 1: An Example to Illustrate the Algorithm Proposed in [7]

As shown in Figure 1, suppose a scan chain with 8 scan cells has an intermittent s-a-0 fault, the algorithm proposed in [7] injects one fault at a time on the faulty scan chain and searches the most matching candidate based on ranking scores. Assuming a scan pattern 11011001 is supposed to be applied to the faulty chain, in Step 1, the loaded pattern will be modified by setting "X" to any possibly corrupted bit. Under the assumption that an intermittent stuck-at-0 fault is at scan cell 4, a loaded scan cell might have incorrect value if it is in the downstream of scan cell 4 and it is supposed to be loaded with a logic "1". For the given example, the loaded scan pattern will be modified by setting cells 0 and 3 to "X". In the next step, 4-value (0,1,X,Z) logic simulation is performed. In Step 3, the captured response will be modified by setting "X" to any possibly corrupted bit during unloading procedure. For the given example, an unloaded scan cell might have incorrect value if it is in the upstream of scan cell 4 and it is captured with a logic "1". Thus scan cells 4, 5 and 7 are set to "X". In the last step, Score A (matching score) and Score B (mismatching score) are calculated for the injected fault candidate. Score A is defined as the ratio of the number of the failure bits covered by the simulated "X"s over the number of total failure bits in this pattern. Score B is defined as the ratio of the number of good bits covered by simulated "X"s over the number of total good bits in this pattern. The procedure iteratively runs for each candidate cell on the faulty chain and reports the cell (or a group cells) that has maximum Score A and minimum Score B as the final candidate fault site.

3.2. Proposed Diagnosis Algorithm for Single Scan Chain with Single Fault

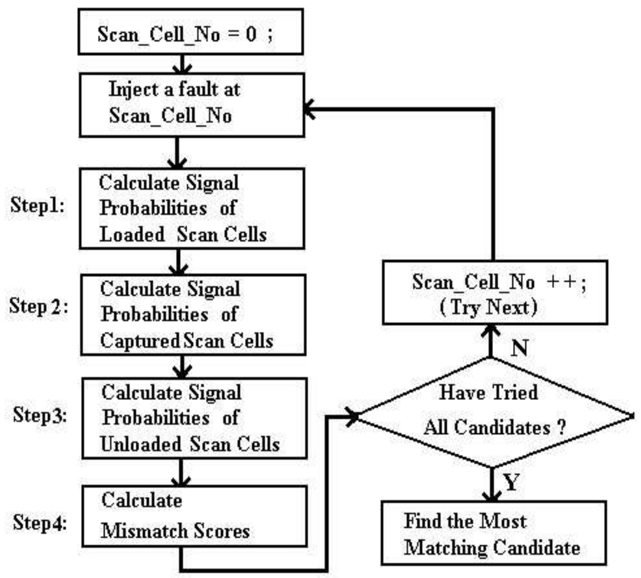

Scan_Cell_No = 0 ;
Inject a fault at Scan_Cell_No
Step1:
Calculate Signal Probabilities of Loaded Scan Cells
Step 2:
Calculate Signal Probabilities of Captured Scan Cells
Step3:
Calculate Signal Probabilities of Unloaded Scan Cells
Step4:
Calculate Mismatch Scores
Scan_Cell_No ++; (Try Next)
N
Have Tried All Candidates ?
Y
Find the Most Matching Candidate

Figure 2: Proposed Algorithm for Diagnosis of Single Fault per Faulty Chain

The flow of the proposed algorithm is illustrated in Figure 2. We assume that an intermittent scan chain fault is triggered with a probability *Prob*. Under this assumption we do not have to perform logic simulation as previously applied in [7].

The proposed algorithm incorporates signal probability calculation in a cause-effect analysis procedure. It injects one fault at a time to the faulty scan chain and searches the most matching candidate based on probabilities. It consists of 4 steps.

Step 1: Calculate *signal probability* for each loaded scan cell.

Signal probability associated with a signal is defined as the probability of the signal being logic "1". Suppose a scan chain has an intermittent fast-to-rise *Hold-Time Fault* and in a pattern two adjacent scan cells (cell $i+1$ and cell i, where cell i is closer to scan out) are loaded with "1" and "0" respectively. During scan chain loading, assuming the "0->1" transition happens at the scan cell where the fault is injected, the probability that the "0" is corrupted to a "1" is the probability that the intermittent fault is triggered, which is *Prob*. Hence after shifting in, the signal probability of loaded scan cell i is *Prob* instead of 0.

This step will calculate the *signal probabilities* for all loaded scan cells based on (1) the known fault type, (2) the location where the fault is injected and (3) the probability of the fault being triggered (*Prob*). Note that it is not easy to determine an appropriate *Prob* since it is per pattern based. In our algorithm this parameter is selected based on one heuristic method that will be explained in Section 4.

Step 2: Calculate *signal probabilities* for scan cells that capture values before shift out.

This step is implemented by extending COP algorithm [10], which is an efficient algorithm for calculating signal probabilities. The calculation of signal probabilities is based on the gate-level structure of the design. Given the signal probabilities of the inputs of a gate and the gate type, the signal probability of its output will be easily calculated. Note that in COP, for each signal, if its probability of being "1" is P_1, its probability of being "0" must be 1- P_1. However in order to apply COP on industrial circuits, we have to extend COP to handle "Z" values. Hence two values are associated with each signal. One is the probability of the signal being "1" (P_1) and the other is the probability of the signal being "0" (P_0). The probability of the signal being "Z" is 1- P_1- P_0. A set of new signal probability calculation rules is added into COP to handle "Z" values.

Step 3: Calculate *signal probability* for each unloaded scan cell.

After the test response is captured into scan chain it will be shifted out. Obviously, some unloaded values might be corrupted if they pass through the faulty cell. This step is similar to Step 1. To give a general equation to calculate the signal probability changes after passing an injected fault, we use intermittent *Hold-Time Fault* as an example, where both "0->1"and "1->0" transitions are possibly corrupted. Assume (P_1^i, P_0^i) and (P_1^{i+1}, P_0^{i+1}) are the probabilities of being "1" and "0" for the captured values at scan cells i and $i+1$ respectively and the probability of the fault being triggered is *Prob*. The "1" probability of the captured value in scan cell i after passing the faulty cell will be:

$$\underline{\mathbf{P}}_1^i = P_1^i * (1-P_0^{i+1}) + P_1^i * P_0^{i+1} * (1-Prob) + P_0^i * P_1^{i+1} * Prob \quad \text{—(1)}$$

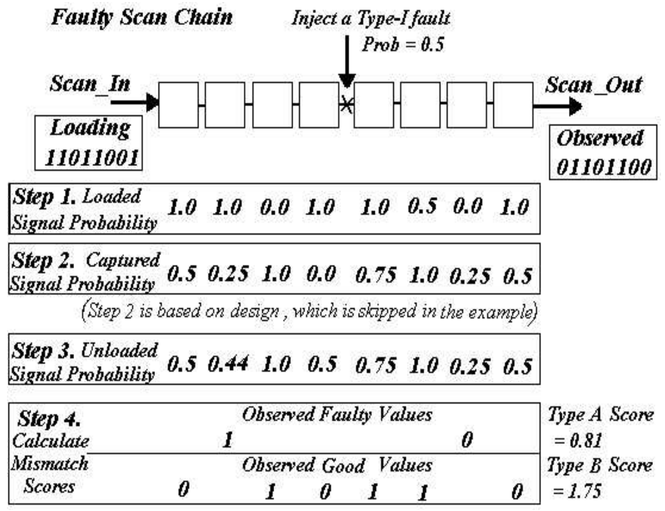
In the right hand side of Equation (1) three items are added up. The first item contributes to the probability that no "1->0" transition happens between the two scan cells so that the "1" probability at cell i is maintained. The second item contributes to the probability that although there is a "1->0" transition between the two scan cells, the intermittent fault is not triggered. The third item contributes to the probability that a "0->1" transition happened between the two scan cells triggered the intermittent fault, which corrupted an original "0" to a "1". Similar calculation is shown in Equation (2) for the "0" probability of the captured value in scan cell i after passing the injected faulty cell. We can deduce the signal probability calculation equations for all other intermittent faults in a similar way.

$$\underline{\mathbf{P}}_0^i = P_0^i * (1-P_1^{i+1}) + P_0^i * P_1^{i+1} * (1-Prob) + P_1^i * P_0^{i+1} * Prob \quad \text{—(2)}$$

Step 4: Calculate mismatching scores for the injected fault.

After the previous three steps, we obtained "1" and "0"probabilities of the values that shifted out of the faulty scan chain. We compare them with the data collected from tester to calculate mismatching scores. Two types of score are calculated in this step. Type-A mismatch scores are calculated at the cycles where a failure is detected from tester. Its purpose is to measure the differences between the calculated values and the observed failures. If "0" ("1") is observed at a failure cycle i, a Type-A mismatch score associated with cycle i is P_1^i (P_0^i). Type-B mismatch scores are calculated at the cycles where no failure is detected from tester. Its purpose is to measure the differences between the calculated values and the observed good values. If "0" ("1") is observed at a good cycle j, a Type-B mismatch score associated with cycle j is P_1^j (P_0^j). For each failure test pattern, we calculate its Type-A (Type-B) mismatch score by adding up all its Type-A (Type-B) mismatch scores at all failing (passing) cycles. For all failure test patterns we calculate the total Type-A (Type-B) mismatch score by adding up the Type-A (Type-B) mismatch score for each failure pattern. The total Type-A and Type-B mismatch scores are used as measures to rank all injected faults. Obviously the smaller of total mismatch scores the more likely that the injected fault is a real fault. In our algorithm Type-A mismatch score is more important than Type-B mismatch score because Type-A mismatch score is calculated based on a complete set of observed faulty bits, whereas Type-B mismatch score is calculated based on a set of observed good bits. Hence we select the faulty site from the candidate list by searching the injected fault with smallest total Type-A mismatch score. The total Type-B mismatch score will be compared only to break a tie. However we may still end up with a range of multiple candidates if both Type-A and Type-B mismatch scores for these candidates are too close to be confidently distinguished. In our experiments, if the difference is within 0.1%, we say they are indistinguishable.

A simple example of intermittent fast-to-rise *Hold-Time Fault* (with *Prob=0.5*) is sown in Figure 3 to illustrate the proposed algorithm. The calculated signal probabilities are listed for each step.



Faulty Scan Chain
Inject a Type-I fault
Prob = 0.5
Scan_In
Scan_Out
Loading 11011001
Observed 01101100
Step 1. Loaded Signal Probability 1.0 1.0 0.0 1.0 1.0 0.5 0.0 1.0
Step 2. Captured Signal Probability 0.5 0.25 1.0 0.0 0.75 1.0 0.25 0.5
(Step 2 is based on design, which is skipped in the example)
Step 3. Unloaded Signal Probability 0.5 0.44 1.0 0.5 0.75 1.0 0.25 0.5
Step 4. Calculate Mismatch Scores
Observed Faulty Values 1 0
Type A Score = 0.81
Observed Good Values 0 1 0 1 1 0
Type B Score = 1.75

Figure 3: An Example for the Proposed Algorithm.

3.3. Diagnose Single Chain with Multiple Faults

After we identify one fault on the faulty scan chain we still don't know whether there are other faults on the same chain. Therefore we will fix the location of the first identified faulty cell and inject another fault at each location other than the fixed faulty location. If injecting the second fault at some locations makes that the combined faulty effect can explain the observed behavior better than single fault, we know a second fault could exist on the same chain. We select the second faulty location by searching the minimum mismatch scores under two faults. This is an iterative procedure to incorporate one more fault at a time and it stops when injecting one more fault doesn't help to reduce the total mismatch scores. Note that if multiple faults are injected on one scan chain the Equations (1) and (2) in Subsection 3.1 should be applied each time a faulty site is passed through during loading and unloading.

3.4. Diagnose Multiple Faulty Chains

If multiple scan chains suffer from the above-mentioned intermittent scan chain faults we have to diagnose the faulty chains one by one. Therefore an appropriate ordering of faulty chains is important. The first faulty chain to be diagnosed should be a chain such that the faults on other chains have the minimum impacts on it. For each faulty chain an *Impact Score* will be calculated as follows. Assume a total of *N* faulty chains in a design. When we calculate the *Impact Score* on one chain, say chain *k*, for each failure pattern, we will set "X" to a scan cell if (1) it is on one of the *N-1* faulty scan chains other than chain *k* and (2) the loaded value at this scan cell could be corrupted. After constraining some loaded bits to "X", 4-value (0, 1, X, Z) good machine simulation is performed. Note that in the simulation we will keep the values loaded into each good scan chain and chain *k*. After the simulation, the "X"s captured into chain *k* must come from the other faulty chains. The lesser number of "X"s captured into chain *k*, the lesser impact caused from other faulty chains. Therefore the *Impact Score* on chain *k* is defined as the total number of "X"s captured into chain *k* when applying all failure patterns to perform the above-mentioned "X"-constrained good machine simulation.

We will first diagnose the chain with the minimum *Impact Score* and assume it is the only faulty chain in the design. After we perform the diagnosis of this faulty chain, the identified faulty location(s) will be fixed on this chain. We repeat the above procedure to re-calculate the *Impact Scores* on all the other *N-1* undiagnosed faulty chains. This time to consider the impact of chain *k* on other chains, we set "X" to the cells on the first diagnosed chain if the cells are in the downstream of the fixed faulty location and are possibly corrupted. To consider the impact of other undiagnosed chains, the same rules of setting "X" used before will apply. The re-calculation is necessary to accurately incorporate the impact of the previously diagnosed faulty chains. We will then diagnose the chain with the minimum calculated *Impact Score* in this run and assume that only two faulty chains are in the design at this moment. The faulty location(s) on the first faulty chain is fixed whereas the iterative fault injections are tried at each scan cell at the second faulty chain. This iterative procedure continues until all the faulty chains are diagnosed.

4. Experimental Results

The proposed algorithm was applied to two industrial designs suffering from intermittent *Hold-Time Fault* (Design 1 and 2). In addition we created a testcase by emulating intermittent *Hold-Time Fault* with a large industrial design (Design 3). Note that the same algorithm can be applied to other fault types. The information about these designs is shown in Table 2. To compare the proposed algorithm with the algorithm previously proposed in [7], the experimental results obtained by the two algorithms are shown in the last two rows of Table 2.

Table 2: Design Information and Experimental Results

	Design 1		Design 2	Design 3
# of gates	147000		325996	2.6M
# of chains	8		10	64
# of faulty scan chains	2		1	1
# of faulty cells per chain	2	1	1	2
Real fault locations	301	57	10	1000
	407			2000
Fault locations reported by the proposed algorithm	300-301	57	10	986-1055
	404-438			2000
Fault locations reported by the algorithm in [7]	300-301	57	10	998-1155
	386-412			2000

As shown in Table 2, Design 1 has two faulty scan chains. One of the faulty chains has two intermittent *Hold-Time Faults* located at scan cells 301 and 407. The other faulty chain has only one intermittent *Hold-Time Fault*, which is at scan cell 57. The diagnosis results obtained by the proposed algorithm report that the first faulty chain has two faults. The first fault is in the range of [300, 301] and the second fault is within the range of [404-438]. The diagnosis results obtained by the algorithm proposed in [7] report that the first fault is in the range of [300, 301] and the second fault is within the range of [386-412]. The rest diagnosis results are also listed in the same format. From the experimental results, we can see that all diagnosis results can report a relatively small range that includes the real faulty cells. The proposed algorithm and the algorithm used in [7] are two different heuristics that produce similar results.

As mentioned before, the probability of a fault being triggered (*Prob*) is an important parameter to get accurate diagnosis result. However it is not easy to determine an appropriate *Prob* because it is per pattern based. (More accurately, it is per cycle based.) Next we will explain how to tune this parameter in our experiment.

If we set *Prob* to 100% it means to treat an intermittent fault like a permanent fault. That is to say, we pessimistically assume that the fault can always be triggered. Although this method can guarantee that the calculated faulty cell ranges are correct, it would drop the diagnosis resolution. Take Design 1 for an example, when we set *Prob* to 100%, we get the diagnosis result showing that one fault is in the range of [255-301] and the second fault is in the range [404-438]. Low diagnosis resolution will be less helpful to locate the real failure sites. If we gradually tune *Prob* from 100% to a number less than 100%, we find the diagnosis resolution will increase before reaching a certain point. After this point the diagnosis range will be incorrect, i.e. not be consistent with the reported range when setting *Prob* to 100%. We used a heuristic method to search this critical point described below.

Initially we set *Prob* to 100% to identify the correct ranges. We tried to tune *Prob* to 90%, 80%, 70% and so on as long as the identified faulty ranges have no conflict with the results when setting *Prob* to 100%. In the experiment of Design 1, when we set *Prob* to 70%, we found the result is conflict with the result when setting *Prob* to 100%. Therefore we set *Prob* to 80%, we can get the first candidate faulty site within [300, 301], which shows an improved diagnosis resolution from 46 to 2. However the resolution for the second faulty site is still remain the same. One can tune the *Prob* with a smaller step such as 5% rather than 10%, but it didn't make any different in our experiments excepted for wasting more CPU time.

5. Conclusions

A new heuristic diagnosis algorithm based on signal probability analysis was proposed to locate intermittent scan chain faults. The proposed algorithm was proved to be effective for large industrial designs with multiple faulty scan chains and multiple intermittent faulty cells on each faulty chain. The results indicated that this algorithm can achieve similar diagnosis resolution obtained by a previously proposed algorithm.

References

[1] R. Guo and S. Venkataraman, "A Technique for Fault Diagnosis of Defects in Scan Chains," Proc. Int'l Test Conference, 2001, pp. 268-277.

[2] K. Stanley, "High-Accuracy Flush-and-Scan Software Diagnostic," IEEE Design & Test of Computers, Nov-Dec, 2001, pp.56-62.

[3] Y. Wu, "Diagnosis of Scan Chain Failures," Proc. Int'l Symp. on Defect and Fault Tolerance in VLSI Systems, 1998, pp.217-222.

[4] S. Narayanan and A. Das, "An Efficient Scheme to Diagnose Scan Chains," Proc. Int'l Test Conference, 1997, pp. 704-713.

[5] S. Kundu, "Diagnosing Scan Chain Faults," IEEE Trans. On VLSI Systems, Vol. 2, No. 4, 1994, pp.512-516.

[6] 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.

[7] Y. Huang, W.-T. Cheng, C.-J. Hsieh, H.-Y. Tseng, A. Huang, Y.-T. Hung, "Efficient Diagnosis for Multiple Intermittent Scan Chain Hold-Time Faults," Proc. Asian Test Symp. 2003, to appear.

[8] R. Wilson, "Statistical techniques attack process variation," EE Times, Mar 6, 2003.

[9] P. Chao and L. Lev, "Down to the wire — requirements for nanometer design implementation," EE Design, Aug., 2002

[10] F. Brglez, P. Pownall, and P.Hum, "Application of Testability Analysis: From ATPG to Critical Path Tracing," Proc. IEEE Int. Test Conf., pp.705-712, Sept. 1984.