Statistical Diagnosis for Intermittent Scan Chain Hold-Time Fault

Yu Huang[1], Wu-Tung Cheng[1], Sudhakar M. Reddy[2], Cheng-Ju Hsieh[3], Yu-Ting Hung[3]

[1] Mentor Graphics Corporation, 8005 S.W. Boeckman Rd., Wilsonville, OR 97070
{Yu_Huang, Wu-Tung_Cheng}@mentorg.com

[2] Department of Electrical & Computer Engineering, University of Iowa, Iowa City, IA 52242
Reddy@engineering.uiowa.edu

[3] Diagnosis Technique & Design Development Division, Faraday Technology Corporation
{cjhsieh, adrian}@faraday.com.tw

Abstract

Intermittent scan chain hold-time fault is discussed in this paper and a method to diagnose the faulty site in a scan chain is proposed as well. Unlike the previous scan chain diagnosis methods that targeted permanent faults only, the proposed method targets both permanent faults and intermittent faults. Three ideas are presented in this paper. First an enhanced upper bound on the location of candidate faulty scan cells is obtained. Second a new method to determine a lower bound is proposed. Finally a statistical diagnosis algorithm is proposed to calculate the probabilities of the bounded set of candidate faulty scan cells. The proposed algorithm is shown to be efficient and effective for large industrial designs with multiple faulty scan chains.

1 Introduction

The development of improved semiconductor process technologies and EDA tools continuously allow the realization of a more complex system on a single die with higher clock frequency and larger system density. However, several new problems, as described below, appear in designing, manufacturing and testing a System-On-a-Chip (SOC) with multi-million transistors using deep-submicron (DSM) technologies.

(1) Signal integrity (SI) and design integrity (DI) issues become major challenges with the evolution to DSM. SI issues include crosstalk, IR drop, power and ground bounce, etc. DI issues include electron migration, hot electrons, wire self-heating, etc [1]. To increase density, interconnects on the chip come closer and are made narrower and thicker [2]. Hence the crosstalk is exacerbated by the increased inter-line capacitive and inductive coupling. Switching signals at a faster rate might cause ground bounce and lowering supply voltages, which lead to decreased noise margins. All the above-mentioned factors would make the signals on the chip more sensitive to unpredicted disturbances from either internal signal changes or external noises. These problems sometimes lead to incorrect functional behaviors, which are called intermittent faults. Unlike the permanent faults that are deterministically modeled by some known fault models (e.g. stuck-at or bridging), the intermittent faults are stochastically observed and difficult to be modeled.

(2) At DSM levels of technologies, statistical timing analysis is the only way to account for wiring delay and process variations [3]. Therefore, instead of thinking that a signal transition happens at a fixed time, we now have to think the transition could happen within a small range of time with an estimated probability distribution density. For example, when we use traditional static timing analysis, we might know that Signal A will have a rising transition at time unit 3. With statistical timing analysis, we might only be confident in saying that with 30% probability the transition happens at time unit 2 and with 70% probability it happens at time unit 3. This trend would change a deterministic digital device into a stochastic computing mechanism.

(3) As the design complexity increases, scan designs are more susceptible to hold-time violations in the scan chains after physical implementation. In DSM designs, it is quite difficult to calculate accurately the wire delay between adjacent scan cells on the real chip. Even with inserted delay element for fixing hold time is sometimes difficult to control the timing precisely after chip fabrication. This type of timing violation in DSM designs may cause the hold-time error when the hold time margin is comparably small. Small hold-time

0-7803-8106-8/03 $17.00 Copyright 2003 IEEE

margins are likely caused by process variation, which is difficult to predict and prevent at the design phase. With shift operation failure, DFT function and fault coverage will be seriously impacted. Therefore, the potential hold-time issue in scan chains could become a major test problem.

(4) In DSM scenario, it is quite likely that the first silicon is designed with errors that escape the static timing verification. Hence the designers are confronted with the problems of silicon debug and diagnosis. The traditional diagnosis techniques should be improved to accommodate the statistical nature of timing and stochastic characteristics of SI / DI issues. The traditional permanent fault models and deterministic test methods cannot capture this uncertainty. We need a new method to debug silicon with low cost, improved design confidence and reduced time-to-market.

In this paper, we propose a novel method called "Statistical Diagnosis" to diagnose intermittent hold-time faults in scan chains. The proposed diagnosis methodology can be easily adopted to target intermittent scan chain faults of other fault types, such as stuck-at or transition faults. Obviously, the permanent faults can be treated as a special type of intermittent faults that have 100% probability of fault occurrence. Hence, in this paper although we only take intermittent hold-time fault as our target, the algorithm can be adopted to diagnose both intermittent and permanent scan chain faults of different types.

The basic idea of our method is to calculate the probabilities of a set of candidate faulty scan cells in a bounded range. The paper is organized in the following manner. In Section 2 we give a brief review of previous works on scan chain fault diagnosis. In Section 3 the intermittent scan chain hold-time fault is discussed. In Section 4 calculation of bounds and statistical diagnosis algorithm are proposed followed by experimental results. Section 5 concludes the paper.

2 Review of Related Work

Full scan design has become the most pervasive DFT technique used in the micro-electronics industry today. Scan chains that function incorrectly will make testing and diagnosis of the complete circuit impossible. Previous works in scan chain diagnosis include but not limited to [4], [5] and [6].

Regarding the scan chain fault types, Wu [4] discussed scan chain hold-time faults and classified them into three types. Type-I hold-time fault captures incorrect data if and only if a "0->1" transition happens at the input of a faulty cell. Type-II hold-time fault captures incorrect data if and only if a "1->0" transition happens at the input of a faulty cell. Type-III hold-time fault happens whenever there is a transition at the input of a faulty cell. We use these three types of hold-time faults in our algorithm as well. Also, to simplify the illustration of the proposed algorithm, we call the transition that activates a certain type of fault *sensitive transition* associated with this type of fault. For example, "0->1" transition is a *sensitive transition* associated with a Type-I fault and "1->0" transition is a *sensitive transition* associated with a Type-II fault. Both "0->1" and "1->0" transitions are *sensitive transitions* associated with a Type-III fault. The difference between the scan chain hold-time faults discussed in this paper and the ones introduced in [4] is the following.

In [4] one type of hold-time fault will always be triggered whenever a *sensitive transition* happens at the input of a faulty scan cell, whereas in our model the fault may or may not be triggered even if the *sensitive transition* happens at the input of the faulty scan cell. This observation is based on data on industrial chips available to us. Whether the fault is triggered or not might also be determined by some other unpredictable circumstances related SI or DI issues. In other words, the hold-time faults discussed in [4] are permanent faults, whereas the hold-time faults considered in this paper are intermittent faults. This will be explained in more detail in Section 3.

In [5] and [6], the authors discussed several fault types including stuck-at fault, transition fault and hold-time fault. In [6], the hold-time faulty behavior is defined as if the faulty latch works like a buffer and the expected values come out of scan chain one clock cycle earlier. The hold-time fault introduced in this paper is completely different with the hold-time fault discussed in [6]. In fact it is similar to the fast-to-rise and fast-to-fall transition faults introduced in [5] and [6], except that the faults considered in this paper are intermittent.

Regarding the methodologies of scan chain diagnosis, in [4] a method to diagnose the faulty chain was proposed by inserting additional set/reset circuitry into the chip. This method may not be acceptable due to its area overhead and performance penalty. In [5] successive fault simulation runs and matching algorithm are proposed. However all scan cells on a faulty chain are candidates, which might reduce the diagnosis resolution. In [6] Guo and Venkataraman proposed an attractive algorithm to compute upper bound and lower bound of the candidate faulty cells. Ranking the suspect scan cells inside the bounded range was used to further improve diagnosis resolution. In the procedure of calculating upper bound and lower bound, they used modified ATPG test patterns to mask the effect of faulty scan cells during the scan chain loading process. This method was called *fully constrained* method that will be reviewed in more detail in Section 4.1 in order to compare with the enhanced upper-bound calculation method proposed in this paper.

3 Intermittent Scan Chain Hold-Time Faults

The timing diagrams for a flip-flop and for two consecutive flip-flops on a scan chain are illustrated in Figures 1 and 2 respectively.

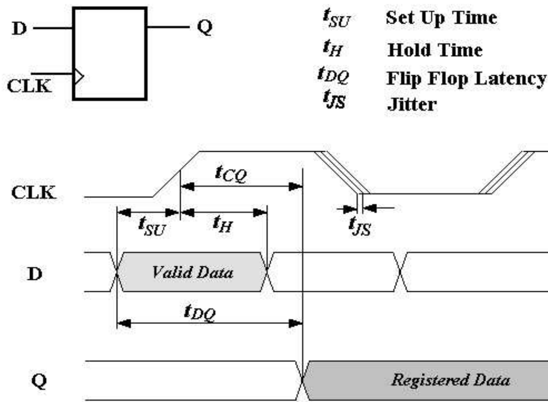

Figure 1. Timing diagram for a single Flip-Flop

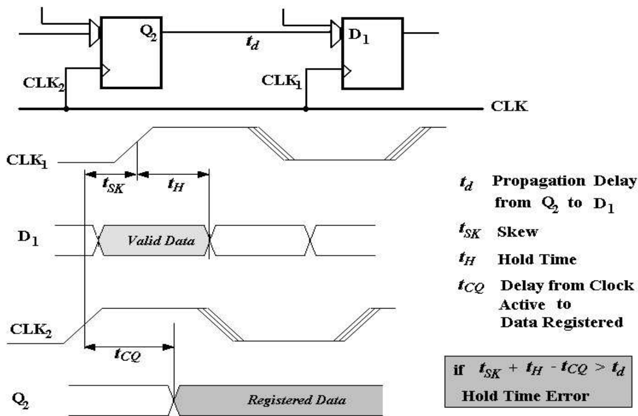

Figure 2. Timing diagram for a Scan Chain

From Figure 1 we observe that the data at the input D of a flip-flop can be correctly registered to its output Q only if it is maintained for at least a specified hold-time (t_H) after the clock is active. Otherwise the registered data may be incorrect. Based on the timing diagram shown in Figure 2, we know that a hold-time fault can be produced if the following two conditions are satisfied.

Condition I: $t_{SK} + t_H - t_{CQ} > t_d$, where t_{SK} is the clock skew between clocks driving two adjacent flip-flops on one scan chain, t_H is the required hold time, t_{CQ} is the delay from activating clock to registering data for the driving flip-flop and t_d is the propagation delay from the output of the driving flip-flop (Q_2) to the input of the driven flip-flop (D_1). In the previously discussed permanent scan chain fault diagnosis [4-6], it was implicitly assumed that if this condition is satisfied, it will always be satisfied regardless of the other signals in the circuit. However, we observed that this is not true for modern designs under the influence of SI/DI issues. For example, if a crosstalk is present between some internal signals in the circuit and the scan path between Q_2 and D_1, it might speedup the signal propagation and hence decrease t_d. Under this situation Condition I, which is not satisfied normally, might be satisfied because of the crosstalk. Therefore a hold-time fault occurs "intermittently" because the crosstalk could intermittently speedup t_d, which is determined by the transition types between aggressor lines and victim lines at a given time. Similar causes might increase t_{SK} and lead to the same result.

Condition II: A *sensitive transition* is supposed to occur at the output of the driving flip-flop between the current shift cycle and the next shift cycle. This condition is a necessary condition to trigger the hold-time faults.

To illustrate the intermittent hold-time error, consider the following example. Suppose a fault-free unloaded (scan out) pattern is 110011001100. A permanent Type-I hold-time fault will lead to the observed unloaded pattern 11**1**011**1**011**1**0, whereas an intermittent Type-I hold-time fault might lead to the observed unloaded pattern 11**0**011**1**011**1**0, in which the first two "0->1" (reading from right to left) transitions triggered the fault, but the last one did not.

Since Condition I might be satisfied due to influence of some unpredictable issues like SI/DI, a scan chain hold-time fault under this circumstance behaves as an intermittent fault. Thus we can assume that the fault, although exists deterministically somewhere on the scan chain, is only triggered with a probability *Prob*. Under this assumption, we do not have to know the physical mechanisms behind the intermittent hold-time fault, which are too complicated to be analyzed easily and efficiently. Consequently, the diagnosis of the faulty site may not point to the exact faulty scan cell. Instead, it gives a range of the candidate faulty cells and calculates the probability $P_{i,j}$ of a hold-time fault that exists between scan cell $i+1$ and i on a scan chain j. To simplify the representation, from now on we just say that the hold-time fault is on scan cell i instead of saying that it is between scan cells i and $i+1$. Note that the scan cell is ordered from scan output to scan input. The scan cell connected with scan output pin is numbered "0". Hence ($i+1$) is the index of the driving flip-flop and i is the index of the driven flip-flop ($i \geq 0$). Our objectives are (1) determining an upper bound and a lower bound on the indices of the candidate faulty cells and (2) calculating $P_{i,j}$ for each scan cell i in the given range in each faulty scan chain j.

4. Proposed Statistical Diagnosis Algorithm

To simplify the discussion of the algorithm, we make few assumptions given below.

(1) Identifying faulty chains and fault types are much easier than diagnosing the faulty site and hence will not be discussed in this paper. Flush patterns used in previous methods [4-6] can be applied to identify the faulty chains and fault types.

(2) Throughout the paper, intermittent scan chain hold-time fault is used as an example to illustrate the proposed algorithm. Other types of scan chain faults, if existing in the design, can be diagnosed by the proposed method (with simple modification).

(3) It is possible that a design has multiple faulty scan chains. However, each faulty scan chain has at most one intermittent hold-time fault. The proposed algorithm will be extended to handle multiple faults on one scan chain in the future.

(4) The hold-time fault can only happen during the scan chain loading/unloading. In other words, the capture is fault-free. The hold-time faults during capture cycles can be diagnosed by using the test generated by the algorithm proposed in [7].

4.1 Enhancement on Upper Bound Calculation

In the proposed algorithm, to calculate an upper bound of candidate faulty cells we use the algorithm proposed in [6] with several enhancements.

We first review the algorithm proposed in [6], which is called *fully constrained* method. Each test pattern is modified by setting all loaded values of the scan cells on faulty scan chains to "X"s. Then logic simulation is performed based on the modified patterns. If there are some known values captured in the faulty chain after simulation, the bounds will be determined based on these known values and the corresponding observed values. For example, assuming that a permanent Type-I hold-time fault exists on one scan chain composed of 16 scan cells. Suppose in one test pattern, the loaded values of the scan cells on the faulty chain are 1001011000110001. Based on *fully constrained* method, the loaded values for this faulty chain are set to 16 "X"s and the loaded values on good scan chains remain unchanged. After logic simulation, suppose the captured value on the faulty chain is X**10**XXXXXXX**10**XXXX, i.e. two sensitive transitions are captured into the faulty chain at scan cells (14,13) and (5,4). Since the loaded "X"s mask the faulty cells, the captured known values ("1"s and "0"s) must be correctly captured into the faulty scan chain regardless whether the loaded values have fault or not. If the observed unloaded values at scan cells that capture a sensitive transition are incorrect ("11"), the permanent Type-I hold-time fault must be in the downstream cells where the sensitive transition is captured. Also, if the observed unloaded values for a captured sensitive transition are correct ("10"), the fault must be in the upstream cells where the sensitive transition is captured. In this example, suppose the tester datalog indicates that a correct transition "10" is observed at scan cells (5,4) but an incorrect transition "11" is observed at scan cells (14, 13), we know that the faulty location must be somewhere between scan cells 13 and 5. This way the upper bound (13) and lower bound (5) of the candidate faulty cells are determined.

In [6], the authors mentioned that the *fully constrained* method could be relaxed by loading some specific patterns. However, we prefer using a fixed ATPG pattern set rather than creating new patterns solely for diagnosis purpose. We improve the *fully constrained* method [6] by applying three techniques in three steps described below. Our objective is to *effectively* set X *constraint*, i.e., we would set as small a number of "X"s as possible and mask all faulty cells in the loading process.

Step 1. Select all scan cells that have active transitions as the initial candidate set, which are possibly corrupted during loading process. In the above-mentioned example, the candidate set includes scan cells (14, 11, 8, 3) given that we knew the chain has a Type-I hold-time fault based on chain flushing pattern responses. Hence, we can mask all possible faulty values by using 4 "X"s instead of 16.

Step 2. The number of possibly to be corrupted scan cells selected in Step 1 could be further reduced by identifying scan cells impossibly to be corrupted, if any in the candidate set, based on the following sub-steps.

Sub-Step (1) To find out whether scan cell, say cell s, is impossibly to be corrupted during scan chain loading for a given pattern, we only set "X" to scan cell s. For the above example, suppose we'd like to examine scan cell 3, we load 100101100011**X**001 to the faulty scan chain and perform logic simulation.

Sub-Step (2) After logic simulation, all scan cells and POs that captured "X"s are grouped into a set, say G_s. We know that if scan cell s is corrupted during scan chain loading, all scan cells or POs in G_s should be observed to have failures for this pattern by tester. If this *per pattern based total match condition* is not satisfied, we know that scan cell s is impossibly to be corrupted during scan chain loading.

Sub-Step (3) In our study, we found that multiple fault masking cannot be ignored in *Sub-Step (2)* given above. A fault on a scan chain could lead to multiple corrupted scan cells during scan chain loading. If more than one scan cells in the fanin cone of a scan cell or a PO has corrupted values, it might produce fault masking, i.e., the scan cell or PO will not capture faulty value due to multiple corrupted cells in its fanin cone. Clearly, if fault masking happens, the *per pattern based total match*

condition is not satisfied. To avoid any chance that a scan cell is mistakenly labeled as an impossibly to be corrupted cell due to the effect of fault masking, we use a relaxed *per pattern based total match condition*, which does not count any cell in G_s that simultaneously satisfies (i) is observed correctly by the tester, (ii) it has multiple fanins from faulty chains and (iii) at least two of these fanins from faulty chains have sensitive transitions.

Sub-Step (4) If for a scan cell *s*, G_s is empty, i.e., its fault effect is not propagated to other places, we still treat *s* as a possibly to be corrupted cell during scan chain loading. This is because its fault effect might appear in the context of multiple faults.

Sub-Step (5) Finally, "X" is set in a scan cell if (i) it is in the candidate set selected in Step 1 and (ii) it is not identified as an impossibly to be corrupted scan cell based on the above *Sub-Steps (1)-(4)*. For example, after *Sub-Steps (1)-(4)*, suppose we know that scan cell 3 is an impossibly to be corrupted cell, we will load to the faulty chain 1X01X11X0011**0**001. After the effective "X" constrained loading, logic simulation is performed and an upper bound is calculated as described in [6].

Step 3. Re-simulate the pattern set for multiple iterations. In each iteration we only set "X" values to the scan cells in the downstream cells of the currently calculated upper bound. The values of the scan cells in the upstream cells of the currently calculated upper bound remain unchanged because these values will not be corrupted during scan chain loading. Therefore, in each iteration we have fewer "X" values loaded into scan cells than in the previous iteration, which might lead to more known values in the captured responses. Hence the calculated upper bound might be updated towards downstream cells. This iteration stops when an upper bound cannot be updated further. For the above example, suppose in the first iteration, we load the faulty scan chain with 1X01X11X00110001 and capture back to this chain X**10**XX11XXX**10**X00X. Also suppose the tester datalog indicates that a correct transition "10" is observed at scan cells (5,4) but an incorrect transition "11" is observed at scan cells (14,13), so we know that the faulty location upper bound is 13. Therefore, scan cell 14, although has a sensitive transition and is a possibly to be corrupted cell, it will not be set to "X" because it is upstream of the current calculated upper bound (13). Hence in the second iteration, we load the faulty chain with 1**0**01X11X00110001 and repeat the procedures. It is possible that the captured response contains more sensitive transitions due to more effective fault mask, e.g. in this iteration, we might capture X10XX1**10**XX10X00X. Suppose the tester datalog indicates that an incorrect transition "11" is observed at scan cells (9,8) then the faulty location upper bound is now updated to 8. More iterations will be tried until no further update of upper bound.

Based on the above-mentioned three steps, we enhance the algorithm proposed in [6] to obtain a tighter upper bound of candidate faulty cells by more effective fault masking. The upper bound calculation method can be applied to either permanent or intermittent faults. However for the intermittent faults considered in this work, a new lower bound calculation method is also used. This is discussed next.

4.2 Proposed Lower Bound Calculation

Unlike the calculation of an upper bound, the lower bound calculation is kind of tricky for intermittent faults. If a captured sensitive transition is observed correctly on tester, it cannot be guaranteed that the intermittent fault must be in the upstream of the cell. The fault might still be in the cells downstream to it, but not triggered in this simulation. To calculate the lower bound we have to use the observed responses from good chains and POs.

First let us assume that only one faulty scan chain exists in the circuit. Multiple faulty chains will be discussed immediately following the case of single faulty scan chain. If erroneous values are observed from A_1, $A_2,\ldots, A_k$, where A_i ($k \geq i \geq 1$) is either a scan cell on good scan chain or one of the POs, the following must be true.

(1) The observed faulty behavior on A_i must be caused by incorrectly loading at least one of a set of candidate scan cells $\{B_{i1}, B_{i2},\ldots, B_{i,si}, (S_i \geq 1)\}$, where B_{ij} ($S_i \geq j \geq 1$) satisfies the following: (i) it is on the faulty chain, (ii) it is within the fan-in cone of A_i, (iii) it has a sensitive transition during scan loading and (iv) it is a possibly corrupted scan cell identified by the procedure described in Step 2 of Section 4.1.

(2) The faulty cell must be in the upstream cells of the last downstream scan cell among the candidate set of $\{B_{i1}, B_{i2},\ldots, B_{i,si}, (S_i \geq 1)\}$, i.e., the scan cell that is closest to the scan out, which is $Min_{j=1..Si}(B_{ij})$.

(3) Lower_Bound $\geq Max_{i=1..k}(Min_{j=1..Si}(B_{ij}))$

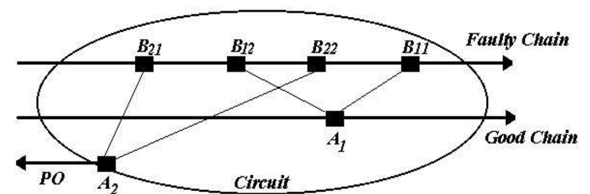

Figure 3. An Example to Calculate Lower Bound for Single Faulty Chain

To illustrate the above observations, let us consider an example shown in Figure 3.

Suppose we observed faults on A_1, which is a scan cell on a good chain and also on A_2, which is a PO. If A_1's candidate set is $\{B_{11}, B_{12}\}$ and A_2's candidate set is $\{B_{21}, B_{22}\}$, we know that the fault must exist in the upstream of B_{11} to cause observed fault at A_1, and in the upstream of B_{22} to cause the observed fault at A_2. Therefore, we determine that the lower bound for the faulty chain is B_{22}.

In case of multiple faulty chains existing in the circuit, we will build a dependency table and calculate the lower bound based on it. We initially build a table as shown in Table I.

Table I: Initial Dependency Table to Calculate Lower Bound for Multiple Faulty Chain

	A_1	A_2	……	A_k	*Lower Bound*
Faulty Chain 1	Min_{11}	Min_{21}	……	Min_{k1}	0
Faulty Chain 2	Min_{12}	Min_{22}	……	Min_{k2}	0
……	……	……	……	……	0
Faulty Chain n	Min_{1n}	Min_{2n}	……	Min_{kn}	0

In the above table, each row represents a faulty chain, each column represents either a scan cell from a good chain or a PO at which an error is observed. The last column maintains the lower bounds for each faulty chain and is initialized to 0 for each row. Min_{ij} represents the last downstream scan cell on faulty scan chain *j* that might cause observed fault on A_i. If there is no candidate set for A_i on scan chain *j*, we use "—" to indicate this situation. The table can be updated according to the following procedures.

Step 1. If there is a Min_{ij} that is larger than the upper bound for the faulty chain *j*, it is obvious that the observed fault on A_i is not caused by chain *j*. Therefore we can use a "—" to replace Min_{ij}.

Step 2. If there is a column, say A_i, that has only one item that is not "—" on one faulty scan chain, say chain *j*, then chain *j* must be responsible for the observed fault at A_i. If the current lower bound for chain *j* is less than Min_{ij}, we update the lower bound for chain *j* to Min_{ij} and delete column A_i from the table.

Step 3. After applying the above steps, it is possible that lower bounds on some faulty chains cannot be updated due to unresolved sources of the observed faults. However, more patterns can be applied to add more columns to the dependency table and Steps 1 and 2 can be repeated for another iteration. The procedure stops when no more update can be achieved for a specified number of patterns.

4.3 Proposed Statistical Diagnosis

After we obtain the upper bound and lower bound of a faulty scan chain, we may end up with a range of candidate scan cells. To improve the diagnosis resolution, ranking the candidate scan cells is necessary. Unlike in [5] and [6], where ranking is based on matching scores calculated by fault simulation, the proposed method applies statistical diagnosis ranking by calculating the probability of each candidate faulty scan cell. We believe that the stochastic nature of intermittent faults is more suitable for applying statistical diagnosis. A probabilistic fault diagnosis was proposed in [8] to target a mixture of fault models in a design. In this paper, however, we propose an entirely different algorithm that specifically targets intermittent faults on scan chains.

We calculate fault distribution probability for each candidate scan cell based on Bayes Theorem [9], which is reviewed below.

$$P(X_k \mid Y) = P(X_k)P(Y \mid X_k) \Big/ \sum_{k=1}^{n} P(X_k) * P(Y \mid X_k)$$

, where $[X_1, X_2, \ldots, X_n]$ is a partition of a set of all possible *n* outcomes of an event *X*. $P(X_k)$ $(k=1..n)$ is typically called apriori probability, which is the probability of event X_k. *Y* is an arbitrary event. $P(X_k|Y)$ $(k=1..n)$ is typically called posterior probability, which is the probability of event X_k given that event *Y* happened.

Let us first define event X_k for our problem. We define event X_k, associated with probability $P(X_k)$, as the fault existing in scan cell *k* of a faulty scan chain *j*. Apriori probability $P(X_k)$ is easily calculated. Assuming that a fault treated as a random variable is uniformly distributed on scan chain *j*, it is obvious that $P(X_k)=1/Length(j)$. Since $P(X_k)$ does not vary with *k*, it can be removed from Bayes Theorem given above. The equation is simplified as below.

$$P(X_k \mid Y) = P(Y \mid X_k) \Big/ \sum_{k=1}^{n} P(Y \mid X_k)$$

We define event *Y* the faulty behavior observed from the tester. $P(X_k|Y)$ represents the probability of a fault existing in scan cell *k* given the faulty behavior observed from tester. According to Bayes Theorem, we have to calculate $P(Y|X_k)$ in order to calculate $P(X_k|Y)$. $P(Y|X_k)$ is the probability of event Y under the condition that a fault is in scan cell *k* of a faulty scan chain *j*. Suppose we have identified the upper bound and the lower bound for the fault site on one scan chain *j*, which are denoted by UP_j and LO_j respectively. Based on the assumption of single fault per chain, $P(Y|X_k)= 0$ if scan cell *k* is out of range $[UP_j , LO_j]$. $P(Y|X_k)$ is measure by the information deduced from the Event Y, which includes the following two conditions:

(1) For all sensitive transitions captured in the downstream of scan cell UP_j on scan chain *j*, we didn't observe any failures caused by unloading.

(2) For any sensitive transitions loaded into the range $[UP_j, LO_j]$ of scan chain *j*, we deterministically know whether this transition is possibly to be corrupted or is impossibly to be corrupted during loading process.

Hence $P(Y|X_k)$ is the probability of simultaneously satisfying the above-mentioned two conditions that event Y must satisfy under the condition that a fault is in scan cell k of a faulty scan chain j. We define $P(Y|X_k) = P_1(Y|X_k)*P_2(Y|X_k)$, where $P_i(Y|X_k)$ corresponds to the probability of condition i.

Calculation of $P_1(Y|X_k)$ and $P_2(Y|X_k)$ is shown in Figure 4. An example is illustrated in Figure 5. To simplify the illustration, we only consider one faulty scan chain j and assume a Type-I hold-time fault in the given example.

$$
\begin{aligned}
P_1(Y|X_k) &= (1-Prob)^u \quad && \text{if } k \in U_Section[u] \\
&= 0 && \text{otherwise} \\
P_2(Y|X_k) &= Prob*(1-Prob)^l && \text{if } k \in L_Section[l] \\
&= 0 && \text{otherwise}
\end{aligned}
$$

Figure 4. Formula of $P_1(Y|X_k)$ and $P_2(Y|X_k)$

In Figure 4, calculation of $P_1(Y|X_k)$ and $P_2(Y|X_k)$ needs explanation of (1) what is *Prob* and (2) what is *U_Sections* and *L_Sections*.

As mentioned in Section 3, we assume that an intermittent fault is triggered with a probability *Prob*, which is calculated by scan chain flushing patterns. For example one chain flushing pattern contains 1000 sensitive transitions on a faulty chain. After the pattern is shifted in and out, suppose 500 sensitive transitions are observed to have failed, then *Prob* = 0.5. In our experiment we use multiple chain flushing patterns and calculate *Prob* for each pattern. The overall *Prob* used in the formula of Figure 4 is the average *Prob* over all the chain flushing patterns used.

U_Section is defined during the upper bound calculation procedure. For each test pattern, first we effectively mask the faulty cells during the loading process (c.f. Section 4.1). Then we clock once and capture the responses back to scan chains. Suppose there are R_j sensitive transitions in the captured response in the range of $[UP_j, LO_j]$ on faulty chain j. We partition the range of $[UP_j, LO_j]$ on faulty scan chain j into $(R_j + 1)$ *U_Sections* that are separated by R_j sensitive transitions. Starting from the upper bound, we number the *U_Sections* 0, 1, 2… etc. In the example shown in Figure 5, the range of $[UP_j, LO_j]$, which is [14,1], is partitioned into 3 *U_Sections: U_Section[0]*= [14,11], *U_Section[1]*= [10,5] and *U_Section[2]*= [4,1]. When the fault is at scan cell k, the probability of the first condition of event Y, which is $P_1(Y|X_k)$, is equal to the probability that the captured sensitive transitions in the upstream of scan cell k are unloaded correctly. $P_1(Y|X_k)$ is determined by which *U_section* scan cell k is located. For the example in Figure 5, we know that two sensitive transitions are captured at (11,10) and (5,4). Suppose cell k is within *U_section[0]*. Since none of the two captured sensitive transitions pass through cell k during unloading process, the two captured sensitive transitions are 100% correctly unloaded. So $P_1(Y|X_k)=1.0$ if cell k is within *U_section[0]*. Similarly, suppose cell k is within *U_section[1]*. Since the sensitive transition captured at (11,10) passes through cell k during unloading process, the probability of the sensitive transition being correctly unloaded is the probability that the fault at scan cell k is not triggered, which is *1-Prob*. So $P_1(Y|X_k)$= *1-Prob if* cell k is within *U_section[1]*. Obviously *if* cell k is within *U_section[2]*, $P_1(Y|X_k) = (1-Prob)^2$, which is the probability that the fault at scan cell k is not triggered for both sensitive transitions passing though.

In general, $P_1(Y|X_k) = (1-Prob)^u$, *if* cell k is within *U_section[u]*, which is the probability of the fault not being triggered by u consecutive sensitive transitions. If we only apply the information collected during upper bound calculation, the scan cell at the upper bound will have the highest probability. To obtain unbiased ranking, the information collected during lower bound calculation is also necessary, which is represented by $P_2(Y|X_k)$.

L_Section is defined after the lower bound calculation. For each test pattern, first we identify the lower bound. Suppose T_j sensitive transitions loaded into scan cells in the range of $[UP_j, LO_j]$ on faulty chain j are identified as impossibly to be corrupted (c.f. Step 2 of Section 4.1). We partition the range of $[UP_j, LO_j]$ on faulty scan chain j into $(T_j + 1)$ *L_Sections* that are separated by T_j sensitive transitions. Starting from the lower bound, we number the *L_Sections* 0, 1, 2… etc. In the example shown in Figure 5, the range of $[UP_j, LO_j]$, which is [1,14], is partitioned into 2 *L_Sections: L_Section[0]*= [1,7], *L_Section[1]*= [8,14]. When the fault is at scan cell k, the probability of the third condition of event Y, which is $P_2(Y|X_k)$, is equal to the probability that (1) the loaded sensitive transition at the lower bound is corrupted and (2) the loaded sensitive

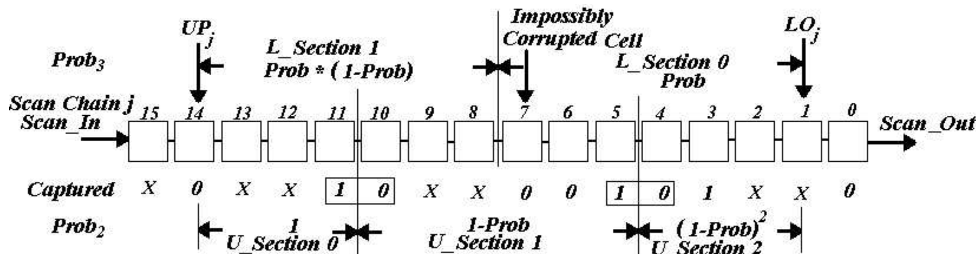

Figure 5. An Example for Statistical Diagnosis

transitions at impossibly to be corrupted scan cells correctly pass through faulty scan cell k. $P_2(Y| X_k)$ is determined by which *L_section* scan cell k is located. For the example in Figure 5, we know that two sensitive transitions are loaded at (1,2) and (7,8). Suppose cell k is within *L_section[0]*. We know that the sensitive transition at (1,2) is corrupted during loading process because its fault effects are observed from good chains or POs. (That is how lower bound is calculated.) The probability of corrupting the sensitive transition loaded into (1,2) by faulty scan cell k is *Prob* if cell k is within *L_section[0]*. That is, $P_2(Y| X_k)=Prob$, which is the probability the fault at scan cell k is triggered. Suppose cell k is within *L_section[1]*. Since the sensitive transition loaded into (1,2) is corrupted while the transition loaded into [7,8] is correctly loaded through cell k, the probability of these two joint events is $Prob * (1-Prob)$. So $P_2(Y| X_k)= Prob*(1-Prob)$ *if* cell k is within *L_section[1]*.

In general, $P_2(Y| X_k)= Prob*(1-Prob)^l$, *if* cell k is within *L_section[l]*, which is the probability of the fault not being triggered by l consecutive sensitive transitions but at least triggered once for the sensitive transition loaded into the lower bound scan cell.

By using the above procedures, we calculate the probability distribution of all candidate faulty cells for each test pattern. Finally we calculate the overall probability of each candidate cell over the entire set of patterns based on the assumption that all test patterns are independently applied. We add up the probability for each candidate scan cell in $[UP_j, LO_j]$ for all the patterns and calculate the average over the total number of test patterns. This way, for each candidate scan cell in the bounded range, a probability is associated with the cell. We rank all candidate cells based on their probability in descending order. The scan cell (cells) with the highest probability is (are) most likely the faulty location. The effectiveness of the proposed method is illustrated through experiments described in the next section.

4.4 Experimental Results

The proposed algorithm is applied to two industrial chips (F1 and F2) that suffered from intermittent hold-time faults. In fact it was these two chips that motivated us to investigate the intermittent scan chain hold-time fault diagnosis. To further measure the effectiveness and diagnosis resolution of the proposed algorithm, we use another large industrial design (M1) to simulate intermittent faulty behavior. We do the simulation by picking one scan chain in M1 and injecting a fault at a scan cell on the faulty chain. The injected fault is a Type-III scan chain hold-time fault, which is triggered by a pre-set probability. A failure file is created to record the observed faulty behavior by using logic simulation. This failure file is an input file for the proposed algorithm, just like a datalog generated from tester. The information about these designs is shown in Table II. The experimental results on F1 and F2 are shown in Table III. The experimental results with various parameters on M1 are shown in Table IV.

As shown in Table II and III, Chip F1 has two faulty chains suffering from intermittent scan chain hold-time faults, whereas Chip F2 has only one faulty chain. By applying the proposed algorithm, for one faulty chain in Chip F1, there is only one candidate faulty scan cell (at cell 57) with the highest probability, which exactly matches the faulty cell (57) on the real chip. That is to say, the diagnosis resolution for this faulty cell is 1. For the other faulty scan chain in Chip F1, a range of three scan cells (from cell 299 to cell 301) has the highest probability. The real faulty site is at cell 301, which is inside the range we obtained. In this case, the diagnosis resolution is 3. For the faulty scan chain in Chip F2, there is a range of four scan cells (from cell 10 to cell 13) that has the highest probability. The real faulty site is at cell 10, which is inside the range we obtained. In this case, the diagnosis resolution is 4. The numbers of the applied scan test patterns (not including chain flushing patterns) and the calculated upper bound and lower bound are shown in Table III.

In F1, however, there is another fault (at cell 407) on the same faulty scan chain where faulty cell 301 is located. Unfortunately this fault site is not covered in the calculated range. It is because that multiple intermittent hold-time faults on one scan chain violate one of the assumptions used in the proposed algorithms. Hence the calculated range may not cover all faulty sites on one scan chain. We are currently working on extending our algorithms to diagnose multiple intermittent hold-time faults on one chain. Other than that, the proposed algorithm seems to be very effective to identify the intermittent scan chain hold-time faults with high diagnosis resolution.

Note that based on our simulation the fault at cell 301 in F1 is responsible for more than 95% observed failures on this faulty chain. Therefore although we only identify one faulty cell on this chain, this one is more important in terms of failure coverage than the other faulty cell on the same chain. But for the purpose of silicon debug, diagnosis of multiple faults on one chain is still necessary.

For the proposed algorithm, we investigate how diagnosis effectiveness and resolution vary with three parameters including (1) different fault sites, (2) the number of test patterns applied and (3) the probability of an intermittent fault being triggered. We used design M1 for this study. For Condition (1), we inject an intermittent Type-III scan chain hold-time fault in M1 at three different locations called Site 1, Site 2 and Site 3. Note that at one time, only one fault is injected in M1. Hence multiple faults are not considered. For Condition

(2), we pick the first 10 or 20 or 30 test patterns from a test pattern set generated for M1 by a commercial ATPG tool. The reason that we didn't use too many patterns to do diagnosis is as follows. All testers have a limited size of memory. A tester will abort testing when the failures in datalog reach its limit. A defect on scan chains would generate much more failures per pattern than a defect not in scan chains. Therefore, for scan chain failure, we may not expect too many patterns applied on tester, which is exactly the case for F1 and F2. For Condition (3), we set *Prob* to 60%, 80% and 100% (i.e., permanent fault) respectively for each run.

For all conditions, the scan cells with the highest probability (within 1% tolerance of difference) calculated by the proposed algorithm does cover the scan cell where the fault is injected. It shows that the proposed algorithm is correct and effective for diagnosing either intermittent or permanent hold-time faults on scan chains under different conditions. However, diagnosis resolution, which is the number of scan cells in the range with the highest probability, varies with different conditions as shown in Table IV.

From the experimental results in Table IV, we have a few interesting observations.

(1) The diagnosis resolution is fault site dependent. The impact of fault location on the diagnosis resolution is large when the number of patterns applied is relatively small. However, with increasing the number of applied patterns, the diagnosis resolutions are good for faults injected at different locations.

(2) Obviously, the number of applied patterns also has impact on diagnosis resolution. The improvement of diagnosis resolution is pattern dependent.

(3) The probability of the fault being triggered has impact on diagnosis resolution as well. The diagnosis resolution is likely to improve when the *Prob* is increased. Therefore the diagnosis resolution for permanent faults (*Prob*=100%) is a lower bound on the achievable resolution. However it seems that the probability of triggering a failure has lesser impact on diagnosis resolution than the number of applied patterns.

Based on the experimental results on M1, the proposed algorithm is seen to achieve good diagnosis resolution for faults at different locations with a small number of patterns.

The proposed algorithm has reasonable computational cost. For chips F1 and F2 used in our experiments, it took less than an hour to compute bounds and probabilities for candidate faulty cells. For each experiment on M1, it took a few hours. Most of the running time is spent on Step 2 of Section 4.1, i.e., to perform logic simulations and to trace back fanin cones to screen out multiple fault masking. Hence, the proposed algorithm is efficient for application to industrial designs with multi-million gates.

5. Conclusions

Intermittent scan chain hold-time fault was introduced in this paper. We proposed a method to calculate an upper bound on the candidate faulty cells, which includes three enhancements on a previously proposed algorithm [6]. We also proposed a new lower bound calculation algorithm by observing the faulty behaviors from good scan chains and POs. A statistical diagnosis algorithm is proposed to calculate the probability of a cell being faulty among the candidate faulty scan cells in the bounded range. The algorithm was applied to industrial designs that suffered from the intermittent scan chain hold-time failures. The diagnosis results match the real defects on chip. The effectiveness and diagnosis resolution of the proposed algorithm were studied by simulating intermittent and permanent scan chain hold-time faults on a large industrial design. The experimental results showed that the proposed method is efficient and effective for large industrial designs with multiple faulty scan chains. Further study will be focusing on diagnosis of multiple intermittent faults on one scan chain.

References

[1] P. Rashinkar, P. Paterson and L. Singh, "System-On-Chip Verification Methodology and Techniques," Kluwer Academic Publishers, 2001.

[2] E. Dupont, M. Nicolaidis, and P. Rohr, "Embedded Robustness IPs for Transient-Error-Free ICs," IEEE Design & Test of Computer,May-June 2002, pp.56-70.

[3] K. Keutzer, ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems, Dec., 2002.

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

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

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

[7] S.M. Reddy, I. Pomeranz, S. Kajihara, A. Murakami, S. Takeoka and M. Ohta, "On Validating Data Hold Times for Flip-Flops in Sequential Circuit," Proc. Int'l Test Conference, 2000, pp. 317-325.

[8] D.B. Lavo, B. Chess, T. Larrabee and I. Hartanto, "Probabilistic Mixed-Model Fault Diagnosis," Proc. Int'l Test Conference, 1998, pp. 1084-1093.

[9] A. Papoulis, "Probability, Random Variables, and Stochastic Process," McGraw-Hill, 1991.

Table II: Information about the Testcases F1, F2 and M1

Designs	# of Simulated Gates	# of PIs	# of POs	# of Scan Chains	Lengths of the longest scan chain	# of Faulty Chains	Real Faulty Sites
F1	147000	95	86	8	725	2	Faulty Chain I: (301, 407)
							Faulty Chain II: (57)
F2	325996	64	51	10	897	1	10
M1	2679502	354	343	64	3184	—	

Table III: Experimental Results on F1 and F2

Designs	# of Applied Scan Patterns	Upper Bound	Lower Bound	The Cells with the Highest Probability
F1	3	Faulty Chain I: 301	Faulty Chain I: 177	(299, 300, 301)
		Faulty Chain II: 57	Faulty Chain II: 0	(57)
F2	16	13	8	(10, 11, 12, 13)

Table IV: Experimental Results on M1

Injected Faulty Site	# of Patterns for Diagnosis	Probability of Triggering Fault	Diagnosis Resolution
Site 1	10	60%	18
		80%	13
		100%	11
	20	60%	7
		80%	4
		100%	4
	30	60%	7
		80%	4
		100%	4
Site 2	10	60%	22
		80%	17
		100%	17
	20	60%	8
		80%	5
		100%	4
	30	60%	7
		80%	3
		100%	1
Site 3	10	60%	15
		80%	15
		100%	12
	20	60%	6
		80%	6
		100%	4
	30	60%	2
		80%	1
		100%	1