# Diagnosis of Defects on Scan Enable and Clock Trees

Yu Huang
*Mentor Graphics Corporation, Marlborough, MA, USA*

Keith Gallie
*LSI Logic Corporation Milpitas, CA, USA*

Scan is the most widely used DFT technique in today's VLSI industry. Mux-DFF and Level Sensitive Scan Design (LSSD) are the most popular scan architectures. For Mux-DFF, when scan enable is set to "1", the scan chain is in shift mode. When scan enable is set to "0", the scan chain is in capture mode. For LSSD, two clocks are used to control the shift. When scan enable or scan clock has defects, it is desirable to locate the defects at logic level by algorithmic techniques to guide failure analysis.

Similar to the defects on other signals, faulty scan enable / clock signals may be caused by numerous types of defects. E.g., a shorted net, an open net or an incorrect timing with respect to clock or scan data stream. The following examples are used to illustrate how to apply various fault models for different defects.

If a scan enable signal is shorted to VCC, only incorrect capturing will result. Scan cells will capture data from the previous scan cell instead of capturing data from system logic. We may use a stuck-at-1 fault model for this scenario. Clearly, the chain integrity test will pass since these patterns don't have the capture operation. The scan patterns would fail and the scan logic diagnosis will be used in this scenario. In rest of this paper, we do not discuss this category of scan enable defects.

If a scan enable signal is shorted to ground, only incorrect shifting will result. So we could use a stuck-at-0 fault model in this scenario. We know that the shift operation sometimes is not correct because instead obtaining the data from the previous scan cell, it will get the data from the system logic during each shift cycle.

If a scan enable signal is shorted to some other part in the design such that its logic value can be toggled, the resultant value will depend on the signal strength of the shorted parts. The fault on scan enable signal may be modeled as a stuck-at-0 (e.g., AND-type bridging), or a stuck-at-1 (e.g., OR-type bridging) or stuck-at-X (i.e., sometimes behaves like stuck-at-0, sometimes behaves like stuck-at-1. e.g. dominant-type bridging). For an open defect, it may also be modeled as one of the three fault models, depending on how the downstream gates of the open site interpret the floating signal. If an open occurs on a stem, different branches may see different stuck-at values based on different thresholds, which may lead to a stuck-at-X fault model.

It is also possible for a scan enable signal to have incorrect timing with respect to the clock or data stream. This could be caused by a design error (e.g., undersized buffers may introduce extra delay) or signal integrity problems (e.g., a crosstalk may speed up the scan enable signal leading to unacceptable skew). Obviously, we need use stuck-at-X fault model for the scan enable timing defects since both shift and capture operations may be incorrect.

Defects on shift clocks in LSSD scan architecture can be modeled in similar way as the fault models for scan enable defects in Mux-DFF scan designs. Therefore in the rest of the paper, we only discuss the diagnosis of stuck-at-0 fault on scan enable signals for Mux-DFF scan designs. Without loss of generality, the proposed algorithm can also be applied to diagnosing defective scan clocks in LSSD scan designs.

If a chain integrity test fails, how would we know if the failure is caused by defects in the scan chains or defects in the scan enable tree? To identify the defects on a scan enable tree, three approaches are proposed in [CRO05]. The first method is to use a scan pattern and vary the timing of the capture (scan enable deassertion) with respect to the clock. The second method is to vary the capture timing with respect to the data stream, which requires the shift operation at very slow frequency to have enough time to deassert / assert the scan enable. The third method is to pad the capture cycle with "dead clocks". These proposed methods may identify timing related defect on scan enable. However, it is not easy to use them to identify other types of defects. That is, if the defect can be modeled as a stuck-at-0, varying capture timing may not help. In this paper, we propose a different software based method that does not require any extra effort manipulating test parameters. First, we assume the defects are on scan chains and use previously published chain diagnosis algorithms [HUA05] to identify the suspect scan cells. Secondly, if there is at least one faulty chain that is modeled with stuck-at-X fault, we attempt to diagnose with stuck-at-0 fault model at scan enable. As we mentioned earlier that the shift operation is incorrect when the scan cell value is obtained from the system logic for each shift cycle, it is very likely we see both stuck-at-1 and stuck-at-0 at scan cells. So stuck-at-X fault model at scan cells is a sign of the stuck-at-0 fault model for scan enable defects.

3-9810801-0-6/DATE06 © 2006 EDAA

The most import idea for diagnosing defects on scan enable tree is to locate the faulty scan cells first, followed by the analysis to find any common scan enable signals that drive these faulty cells. Using stuck-at-X model, we cannot locate all faulty cells. However, we may identify the boundaries of two special faulty cells: the faulty cell located closest to the scan input (called input-end faulty cell) and the faulty cell located closest to the scan output (called output-end faulty cell).

First, we use a binary search algorithm to identify the lower bound for the input-end faulty cell. Suppose the design has only one faulty scan chain with 200 scan cells. We partition the chain into two segments at the middle (cell 100). Note cell 0 is the cell at scan output end. For each scan pattern, we modify the loading values on this faulty chain such that the loading values at cells at downstream of cell 100 are X. Then we capture the test responses based on the modified loading values. Next, we compare the simulated results against the observed results on *good* chains. Note that the failure bits on good chains are the results of incorrect shifting on faulty chain. If at least one cell on good chain shows a conflict (simulation is 1/0, observe 0/1), we know that the at least one cell in the range [199, 100] has incorrect loading values. We then move our partition point to cell 150 and repeat the above binary search. If there is no conflict in the upper section of the chain [199,100], we move the partition point to cell 50 and repeat the above binary search. The search will converge on a cell $L$ such that setting the partition point at cell $L$ leads no conflict at good chains whereas setting the partition point at cell ($L-1$) (i.e. the adjacent downstream cell of $L$) leads to conflict at good chains. We know that the cell $L$ has an incorrect loading for this conflict scan pattern, which makes cell $L$ a lower bound for the input-end faulty cell.

Secondly, we use faulty chain simulation to identify the upper bound for the output-end faulty cell. This is same as the upper bound calculation method for single fault proposed in [GUO01]. Let us review it by using an example as follows. We first modify all loading values to all Xs for each scan pattern. Then we perform simulation to get captured values. Suppose a scan pattern captures a known value ("0" or "1") at cell 100 on the faulty chain and this known value is observed as a failure on the ATE. This implies that there must be a fault in the range of [0, 100] on this faulty chain because the value captured at cell 100 must be corrupted during the unloading procedure. Therefore cell 100 is an upper bound for at least one faulty cell based on this pattern. If we find this scenario happened in multiple patterns, we select the minimum upper bound among these patterns as the upper bound for the output-end faulty cell.

At this point, we obtain two ranges per faulty chain that can be modeled with multiple stuck-at-X faults. If we assume there is one defect on scan enable tree that causes the defective chains, the scan enable signal must drive at least one scan cell per range. Our purpose at the final stage is to locate this faulty scan enable signal. It is done by the following four steps:

*Step 1:* We perform backward trace on the scan enable tree from the scan cells in each range. For scan cell $i$, we find a set of signals, denoted as $SE_i$, on the scan enable tree, which drives the scan enable pin of cell $i$. Obviously, if a node drives a scan cell, all its parent and ancestor nodes on the scan enable tree are the drivers of this scan cell as well.

*Step 2:* Find the union of the suspect nodes on scan enable tree of all cells in the same range. For a range $r$, we denote this union as $U_r$ and $U_r = \cup_{(\text{all cell } i \text{ in range } r)} SE_i$.

*Step 3:* Find the intersection among all the unions identified in the previous step. We denote this intersection as $I$ and $I = \cap_{(\text{all range } r \text{ on all faulty chains})} U_r$.

*Step 4:* We order the nodes in set $I$ based on its depth on the scan enable tree, in decreasing order. Then we perform forward trace from each node in set $I$ to scan cells. The forward trace is performed one node at a time according to its order in $I$. If a node also drives at least one scan cell on a good chain, it is not a suspect. So we drop this node from $I$. Meanwhile, we drop all its parent and ancestor nodes on the scan enable tree if any of them are in $I$ as well. The searching order of $I$ avoids any unnecessary forward tracing from higher level nodes. After we apply this procedure to all nodes remained in set $I$, we will find all suspect scan enable nodes that satisfy the conditions (a) drive at least one scan cell per range for all the ranges we identified on faulty chains and (b) don't drive any cell on good chains. These remained nodes in $I$ will be reported as suspect defect locations in the scan enable tree.

The proposed algorithm is implemented in a commercial EDA tool. It has been applied to two industrial VLSI circuits. Each test case has one signal on scan enable tree shorted to ground, which is introduced via a Focused Ion Beam (FIB) machine. The defective chips are tested with both chain integrity patterns and scan patterns. The failing data is collected from ATE and read by the diagnosis tool. The tool reported suspects, which include the injected defects. The diagnosis resolutions are good: in one case the tool reported two suspect signals and in the other case, the tool reported 4 suspect signals. The proposed diagnosis algorithm is proved to be an effective and efficient way for diagnosing this special type of defects.

## REFERENCES

[CRO05] A. Crouch, "Debugging and Diagnosing Scan Chains," EDFAS, Vol. 7, Feb., 2005, pp 16-24.

[GUO01] R. Guo et. al., "A Technique for Fault Diagnosis of Defects in Scan Chains," ITC, 2001, pp. 268-277.

[HUA05] Y. Huang et. al., "Using Fault Model Relaxation to Diagnose Real Scan Chain Defects," ASPDAC 2005, pp.1176-1179.