On the Relationship between Stuck-At Fault Coverage and Transition fault Coverage

Jan Schat
NXP Semiconductors
jan.schat@nxp.com

Abstract

The single stuck-at fault coverage is often seen as a figure-of-merit also for scan testing according to other fault models like transition faults, bridging faults, crosstalk faults, etc. This paper analyzes how far this assumption is justified.
Since the scan test infrastructure allows reaching states not reachable in the application mode, and since faults only detectable in such unreachable states are not relevant in the application mode, we distinguish those irrelevant faults from relevant faults, i.e. faults detectable in the application mode.
We prove that every combinatorial circuit with exactly 100% stuck-at fault coverage has 100% transition fault test coverage for those faults which are relevant in the application.
This does not necessarily imply that combinatorial circuits with almost 100% single-stuckat coverage automatically have high transition fault coverage. This is shown in an extreme example of a circuit with nearly 100% stuck-at coverage, but 0% transition fault coverage.

1 Introduction

Customers are constantly asking for lower and lower DPPM (Defective Parts per Million) levels. This requires very elaborate production tests. Besides single stuck-at (SSA), many non-SSA fault models have to be considered like transition faults, path delay, bridging, and crosstalk. With smaller feature sizes, the percentage of stuck-at faults is declining along with the confidence in the SSA-targeting scan patterns as the major gatekeeper for logic faults. Thus more attention is paid to coverage of patterns designed for scan testing according to other fault models.

While SSA fault coverage usually reaches 99% or more in today's designs and today's ATPG engines, coverage figures for transition faults or bridging faults as reported from the ATPG engines are much lower, typically in the range of unsatisfying 70% - 95% [1] (for a remarkable 99% see [2]).

While it is clear that high SSA coverage *usually* implies high coverage for non-SSA scan tests, these questions often arise:

a) Is a transition fault pattern with 80% coverage so much worse than an SSA pattern with 99% coverage?
b) Does high SSA coverage *always* imply high coverage for non-SSA scan tests?
c) Does 100% SSA coverage always lead to a certain minimum coverage for non-SSA scan tests?

This paper mainly addresses these three questions on the transition fault model as an example of a non-SSA fault model. To answer these questions, one has to look closer at the set of targeted faults, also known as 'fault universe':

There is always a certain number transition faults, which can be detected in test, but not in the application [1,3,4]. This is because some states are not reachable in the application; also, not all transitions from one reachable state to another can occur in the application. The faults that are only detectable in these transitions are thus undetectable and hence not noticeable in the application. It should be thoroughly considered if these faults should really be included into the set of targeted faults.

Fault coverage is defined as the number of detectable faults in this set, divided by the number of all faults in this set. Thus, if this set contains faults which are undetectable in the application, the calculated coverage will be misleadingly low.

The rest of the paper is organized as follows: Chapter 2 summarizes the reasons for not having 100% SSA coverage. Chapter 3 presents a small example to motivate the distinction between different sets of faults, also answering the previous question a). Chapter 4 summarizes considerations whether ICs with defects not disturbing the application mode should be shipped or not. Chapter 5 shows the example that even close-to-100% SSA coverage may mean 0% transition fault coverage, thus answering the previous question b). As the main part of this paper, Chapter 6 presents proof that 100% SSA coverage means that a transition fault test pattern set can be created with 100% coverage for those faults that can disturb the application; this answers the previous question c). Finally, Chapter 7 concludes.

978-3-9810801-5-5/DATE09 © 2009 EDAA

2 Factors reducing the fault coverage

A 100% SSA fault coverage is rarely achieved in combinatorial logic in real-life ICs. This is mainly because of three reasons:

- Netlist redundancy (including unused gate outputs, constant inputs etc.). Redundancy is usually unwillingly added when compiling RTL code to a gate level netlist, typically accounting for some 0.1 - 1.0 % of all faults being undetectable.
- Outputs of RAMs, mixed-signal blocks etc. often have a state which is unknown to the ATPG engine. The same applies to signals crossing a clock domain, i.e. going from one island of synchronicity to another. To ensure complete testability, these signals should only go to a scan testable flip-flop, not to combinatorial logic.
- Some ATPG engines abort the calculation of a test pattern for a targeted fault when a specified calculation time is exceeded. This is no longer an issue in today's ATPG engines, however, since SAT solvers are available; those calculate scan test patterns for each detectable fault [5].

Good design practice can lead to 100% SSA coverage, but approximately the last half percent of the coverage can often only be achieved manually by test point insertion and redundancy elimination.

3 Example: Synchronous counter

To illustrate the different set of targeted faults, a simple example follows (see [1] for another).

Consider a synchronous counter that divides a clock signal by three. It has three different states (here simply called 0, 1 and 2) and must hence have two flip-flops. A circuit with two flip-flops has four states, however (the mentioned states 0,1,2, and an additional state called 3), of which states 0,1 and 2 are used, and are thus reachable in the application, and state 3 is not used and is thus unreachable in the application. Of course, on power-up, each of the states may occur. The production test must ensure that from the possible initial state 3, the circuit can reach state 0 upon reset (also called initialization), and that from the other states the circuit can reach the next (Figure 1).

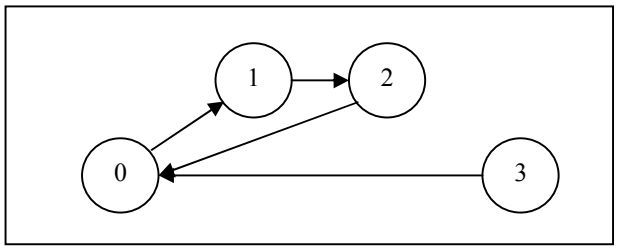

Figure 1: States and transitions in a synchronous counter

Which transitions between two states should be considered for the set of targeted transition faults? An ATPG engine could calculate all possible transitions from each of the states 0…3 to each of the other states, thus 3*4=12 transitions.

Since only states 0, 1 and 2 are reachable in the application and are thus relevant, however, the number of transitions considered for testing could be reduced to 2*3=6 transitions.

In the application, however, only three transitions can occur, namely 0 → 1, 1 → 2 and 2 → 0. Only during the reset, also the transition 3 → 0 can occur.

Thus, we to choose between four possible sets of transitions to consider:

a) All 12 transitions from each state to another state;
b) All six transitions from each reachable state to the other reachable state;
c) All four transitions which can occur in the application mode, including the reset;
d) All three transitions in application mode, excluding the reset.

While it seems reasonable to only consider the fault set d), the problem is that the ATPG engine usually does not know if a state is reachable in the application or not (see [4] for an exception). This does not mean it calculates all possible transitions according to a), however: the launch-on-capture architecture only permits transitions *to* a reachable state (no matter if the state *from* which the transitions occurs, is reachable or not), which in this example (but not always) is identical to the fault set c).

In this example, an ATPG engine which calculates a pattern set for transition faults using launch-on-capture will have all 12 transitions according to a) in the fault set, while the launch-on-capture architecture only allows for testing the four transitions according to c). The engine will thus report a transition fault coverage of 4/12=33.3% - even though all transition faults can be tested that can occur in the application. This low reported coverage is likely to concern the test engineer, because he or she will think the majority of faults of an important fault class remains untested.

This example shows it is very important to note which set of faults the number of faults refers to. For a more detailed discussion, see [1,3,4]; for the importance of considering reset states, also [6].

4 Uncritical faults and Reliability

It is sometimes argued that ICs with non-application-critical faults should be not shipped, even though they are fully functional. The main concern is reliability: the physical defect causing this fault might worsen in the course of time and lead to an application-relevant fault. There are good reasons, however, to ship those ICs anyway:

- Any kind of defect is seen minimally critical in RAMs: repair using spare rows and columns is a generally accepted (and often the only possible) way for manufacturing large RAMs. After repair, of course the defect remains, but is made redundant and undetectable. No noticeable influence on reliability was reported [7]. Thus the same is likely to apply to combinatorial logic.
- Transition faults in non-application paths should not be considered, because these paths are usually longer than the optimized application-critical paths. Furthermore, current consumption could be higher than designed, leading to higher IR voltage drops and thus to overtesting, i.e. throwing away parts that are fully functional, but failing the test only because of the IR drop in those non-application-mode patterns [1].

5 Close-to-100% SSA coverage, yet 0% Transition fault coverage

To show that even with a nearly 100% SSA coverage the transition fault coverage can be very poor, the following example is presented:

Tie-off cells are used to deliver a constant logic value – either 0 or 1. Usually they are made scannable so that in the scan test, they can deliver both values. For this example, we consider a cell named 'misdesigned scannable tie-off cell' (MSTO cell) as shown in Figure 2. This cell is indeed controllable in the first normal mode cycle, but, by misdesign, outputs an undefined signal in the second normal mode cycle. In all following cycles, i.e. in the application, it outputs a constant 0 (Figure 3).

This MSTD cell is therefore functional for stuck-at scan test, functional for the application, but fails badly in the transition fault test.

Imagine now that the output signal of this cell is used to XOR the input signal of all scan flip-flops (SFFs) (Figure 4). In the first normal mode cycle, a defined signal is output from the MSTO cell, so that the input of the SFFs is also defined. This also means that the complete combinatorial logic (marked by the dashed box) is nearly 100% SSA testable. The only untestable node is the 'X' node of the MSTO cell.

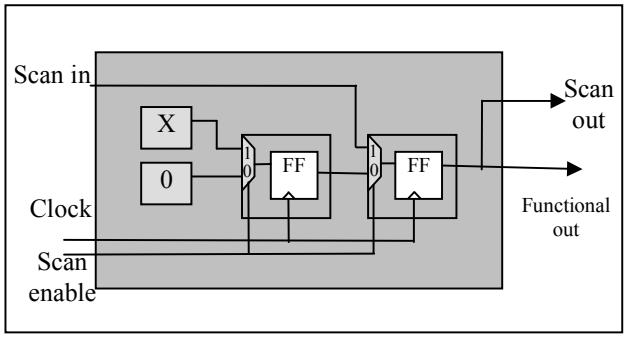

Figure 2: a misdesigned scannable tie-off (MSTO) cell

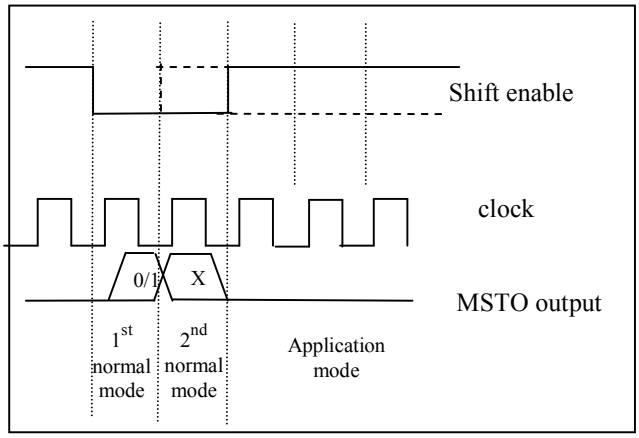

Figure 3: Output of the MSTO cell in the transition fault test (dashed lines for shift enable: stuckat-scan test and application mode)

In the second normal mode cycle, however, all signals from the combinatorial logic to the scan flip-flops are XORed with the unknown signal of the MSTO cell, thus also the output of the XOR gates is undefined.

In other words, a completely stuck-at testable combinatorial logic becomes untestable for transition faults due to this MSTO cell. Of course, circuits like this are never purposely designed. This example was chosen to illustrate that an extremely high SSA coverage (just one node of the MSTO cell is not SSA testable) can even lead to 0% transition fault coverage.

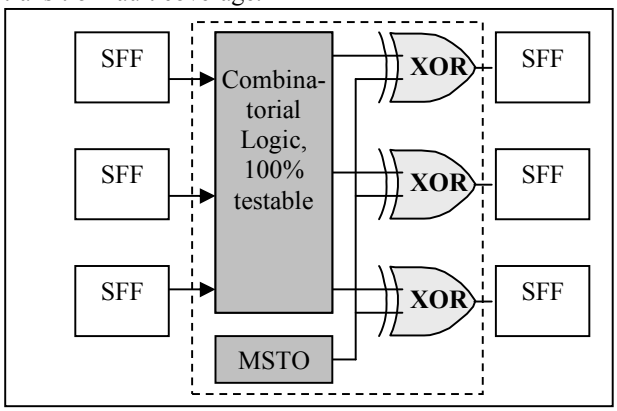

Figure 4: MSTO cell in a design, spoiling the transition fault coverage

6 100% SSA Coverage leads to 100% transition fault coverage

While in the previous chapter it was shown that even a close-to-100% SSA coverage means nothing for the transition fault coverage, it can be proven that exactly 100% SSA coverage means 100% coverage for all transition faults which are relevant in the application. This, to the best of the authors' knowledge, has never been proven before.

Cororally:
The combinatorial logic has the same behavior (state table, or Boolean expression) in the normal mode(s) of a scan test, as in the application mode

Proof:
Since 100% SSA fault coverage is assumed, there cannot be constant signals as input signals to the combinatorial logic. Any difference in behavior between normal mode and application mode would be caused by such a constant signal. A contradiction. □

Assumption:
For each combinatorial logic with 100% SSA fault coverage, a transition fault pattern set can be generated which has 100% coverage for all application relevant transition faults.

Proof:
We consider the launch-on-capture architecture only.
Without loss of generality,

- We only consider transitions $0 \rightarrow 1$.
- We only consider one clock domain.

Since we have 100% stuck-at fault coverage, for each node N_i there is at least one state, named S_2, with:

- $N_i = 1$
- N_i is observable at the outputs of the combinatorial logic, i.e. at the input of the scan flip-flops.

This state S_2 should be the second normal mode state of a transition fault test. Can this state S_2 in a transition fault test (launch on capture) be derived from another first-normal mode state, named S_1, and cause a transition $0 \rightarrow 1$ at N_i? To answer this question, we consider all states, named $\{S_{2A},..., S_{2X}\}$, for which both above conditions are met:

Case 1: *none* of the states $\{S_{2A},..., S_{2X}\}$, fulfils the additional condition:
There is at least one state S_1 with

- $N_i = 0$
- In the transition fault test (and in the application) S_1 is transformed to S_2 in the following cycle.

In this case the transition N_i: $0 \rightarrow 1$ is not observable in the application, and thus a failing transition is a redundant fault.

Case 2: *at least one* of the states $\{S_{2A},..., S_{2X}\}$ fulfils the additional condition:
There is at least one state S_1 with

- $N_i = 0$
- In the transition fault test (and in the application) S_1 is transformed to S_2 in the following cycle.

In this case, the transition N_i: $0 \rightarrow 1$ is relevant in the application, and is testable in a transition fault test by the two states S_1 and S_2.

The above is valid for each node N_i. Thus for each node N_i the transition fault $0 \rightarrow 1$ is either redundant, or detectable. Thus the test coverage, considering only faults relevant in the application, is 100% for the whole combinatorial logic. □

7 Conclusion

Stuck-at coverage is a figure of merit, also for the coverage of other scan-based tests for bridging, path delay and other faults. The relationship is loose, however; in extreme cases, 100% stuck-at coverage can mean 0% transition fault coverage, as was shown in an example. On the other hand a proof was presented, that 100% stuck-at coverage also means 100% coverage for those transition faults that can be detected in the application. A good ATPG engine in this case can calculate a transition fault pattern set, which indeed covers all transition faults which can occur in the application.

The reported coverage will be much lower than 100%, however, because also non-relevant, undetectable transition faults are in the list of targeted faults.

100% stuck-at fault coverage is indeed reachable when using today's design tools, ATPG engines and some manual optimization.

This astonishing fact is important for assessing the usually reported low coverage in non-stuck-at scan tests like transition faults. Since their coverage for application-relevant faults is comparable to the coverage of stuck-at tests, this fact helps gain confidence in these tests, reassure test engineers and customers, and helps explain why relatively low DPPM figures are achieved with apparently low transition fault coverages.

References

[1] J. Rearick, "Too much fault coverage is a bad thing", International Test Conference 2001, p. 624 -633

[2] M. P. Kusko, B. J. Robbins, T. J. Koprowski, and W. V. Huott, "99% AC test coverage using only LBIST on the 1 GHz IBM S/390 zSeries 900 Microprocessor", International Test Conference 2001, p. 586-592

[3] I. Pomeranz and S. M. Reddy, "Expanded definition of functional operation conditions and its effects on the computation of functional broadside tests", VLSI Test Symposium 2008, p. 317-322

[4] I. Pomeranz, "On the generation of scan-based test sets with reachable states for testing under functional operating conditions", Design Automation Conf. 2004, p. 928 – 933

[5] T. Larrabee, "Test pattern generation using boolean satisfiability", IEEE Transactions on computer-aided design, Vol. 11, No. 1, p. 4-15, 1992

[6] M. Abramovici and P. S. Parikh, "Warning: 100% Fault Coverage may be misleading!!", International Test Conference 1992, p. 662 – 668

[7] T. S. Barnett, M. Grady, K. Purdy, and A. D. Singh, "Redundancy implications for early-life reliability: experimental verification of an integrated yield-reliability model", International Test Conference 2002, p. 693-699