output logic pin q. Output pin qn is just q inverted.

This simple example illustrates the essence of scan testing. For a fault on an internal net to be detected, the ATPG tool must be able to control the value to put on that net as well as to propagate that value to either a primary output pin or the scan output pin for observation. If a fault is not detectable, then one or both of these requirements is not met.

Over the years, many DFT rules have been developed for high fault coverage; if they are followed, scan-vector generation should be straightforward and very fast. However, that is a big if—many designers still do not pay enough attention to them. As a result, the initial fault coverage averages about 75 percent to 85 percent, less than the required 95 percent or higher. Fault-coverage-analysis studies why a fault is not detectable and tries to find ways to detect it. However, this is usually a mammoth challenge because uncovered faults can number in the thousands even for so-called small designs. Our task is to study every one of them and find ways to cover them in days instead of months!

In this article, practical methods are presented to systematically analyze stuck-at faults not covered by full scan, with real-world design examples. Discussion will be centered around important DFT rules that, if followed, greatly increase fault coverage.

Fault-coverage analysis is based on single stuck-at faults. This means that if a pin of an internal gate is stuck at 0 or 1, the scan vectors can propagate this fault to the outputs for observation. This is not the same as design verification test (DVT), which studies the design percentage that has been functionally verified and exposes flaws that do not meet specifications. For example, if the OR gate in Figure 1 is mistakenly replaced by a NOR gate scan vectors cannot reveal this design flaw. Also, fault coverage is not toggle coverage; the latter is part of DVT and involves running simulation on the design while a toggle tool tracks the number of times each line of RTL code is exercised. Its main purpose is to reveal the logic that has not been "touched" by logic verification so that new tests can be written to verify it. Finally, scan testing does not check circuit performance (timing), which must be studied with static/dynamic timing analysis.

### DFT rules

A common ASIC design methodology starts with an architectural specification for its functions and design partitions. This is followed by implementation in a hardware description language, such as Verilog and VHDL, or a general-purpose programming language, such as C and C++, to generate a design netlist. This netlist is then turned into a gate-level design by synthesis. Most synthesis tools are coupled with ATPG tools so that testability problems can be analyzed during synthesis instead of afterward. Another, older, approach is to postpone testability study until synthesis completes. Either way, one should face the same fault-coverage problems.

A design has a fixed number of possible faults, since each net can have at most two faults: stuck-at-0 and stuck-at-1. Fault coverage is defined as follows: Fault Coverage = Detected Faults/Total Faults.

It is obvious that to get high coverage, the numerator must be as large as possible since the denominator is fixed. Commercially available ATPG tools can generate scan vectors for very large designs in 30 minutes or fewer. However, they cannot provide a high coverage unless a design follows DFT rules.

After initial testability study (during or after synthesis), every possible fault in the design should fall into one of the following three categories:

- **Detected Faults**: should be the majority (75 percent to 99 percent) of Total Faults. The test tool detects them within the given constraints, such as run-time limits. If the percentage is 95 percent or more, there is still a need to perform fault-coverage analysis.
- **Undetected Faults**: should be a very small part of Total Faults, assuming the design is suitable for full scan testing. Almost all should fall into the type of ABORTED, which means the test tool has aborted trying to detect these faults because of run-time limits. Some fault loca-