

What Can IC Test Teach System Test?

Scott Davidson
Sun Microsystems Inc.
901 San Antonio Road
Palo Alto, CA 94303

Once upon a time there were only functional tests. Smart people, people who understood how the product worked, tried to cover all parts of the spec with their test. They used checklists to track their progress, but the ultimate measure of their success was how well the product did after being shipped. If it failed in the field even though their test passed, they tried to diagnose the problem and improve their test, but often this was hard to do.

The biggest problem they faced was that the increasing complexity of the products they were dealing with meant that test writing took longer, escape rates were higher, and diagnosis was harder. *Ad hoc* techniques helped some, but a quality nightmare seemed inevitable - especially when customers started demanding products that worked right out of the box.

I'm talking about IC test 25 years ago, but I could be talking about system test today. How did we get out of the fix I described above? My contention is that things got better when we started measuring fault coverage.

Engineers writing tests manually cover all the cases they think of, and measure themselves on that basis. When you fault simulate these tests, coverage for all the cases not thought of is obtained. Faults only detectable through non-intuitive tests are modeled, as well as those which are more obvious. Most importantly, a single number, understandable by even the most pointy-haired of managers, is produced. We realize today that the accuracy of stuck-at coverage is questionable, but there is no doubt that a 99% coverage test is better than an 80% coverage test.

Once fault coverage is known, there is a drive to improve it. This can be done either by writing more tests or by changing the design style to let a tool, or the silicon itself, create the test for you. Complexity and time to market pressures led to the acceptance of scan, with the result that today our IC quality is excellent, especially when considering the size of the average ASIC.

What lessons does this story have for system test? Today, the coverage of a system test is a great unknown. Yes, faults can be inserted within a system, the test run, the results determined, and a rough coverage obtained. However, one used to dealing with ten million stuck-at faults in a single ASIC tends not to have a great deal of confidence in the cov-

erage obtained by inserting a few hundred faults. Physical fault insertion can prove that a test is deficient, but never that it is good. The types of defects that can be inserted using physical fault insertion are limited. Stuck-at faults are probably even a worse model for evaluating a system test than they are for an IC test.

Not much has happened in system test for quite some time. Physical fault insertion is hardly new - I know someone whose first job at Bell Labs was doing this for telephone switches - in 1960. Built-in test has helped, as have various diagnostic registers that can take snapshots of the state of a system upon failure. We do know how to recover from failures, but we do not know whether a particular test is going to catch a failure. To do that, we need some effective means of getting coverage.

What would the ability to determine coverage mean for system test? First, test writing effort could be targeted to areas of weakness in a test, maximizing its effectiveness. We can ensure that there are not holes in the test, with sections of the system being covered lightly if at all. We can determine structures which resist testing, and suggest design changes or DFT features that can be used to improve coverage. This would certainly be an iterative process, extending over several product generations. We should never expect anything as radical as full scan, or ATPG for a system, though. Finally, we should be able to model many types of defects, both to make sure they are covered, and also to improve system diagnostics through defect insertion. There has been some work in this area but not nearly as much as is required to bring the ability to compute system test coverage anywhere near the ability to determine IC test coverage.

Determining coverage will be no simple task, but with high level modeling languages, powerful simulators and multi-processor compute servers it should be possible. A simulation based approach would allow test writing and coverage analysis.

The features, methods and procedures required to improve the quality of system test are impossible to predict now. We can be sure, however, that with a clear system level quality and coverage metric, we will begin to move in the right direction.