Test Points are Trending


In New test points slash ATPG test pattern count, I described a new type of test point technology used with scan compression for device testing. The key benefit of using test points with embedded compression is a marked decrease in test pattern count, along with the related reduction in test cost and time.

Test engineers are adopting EDT Test Points rather quickly because they are easy to add to an existing methodology and give a clear advantage in test compression and time. But a question has come up that deserves a detailed answer: are EDT Test Points acceptable for use during at-speed test?

The answer is yes, but how and what types of Test Points to use during at-speed test depends on several factors. There are some concerns that Test Points actually interfere with at-speed paths and therefore shouldn’t be used. One thing to consider when deciding to enable EDT Test Points during at-speed testing is what types of defects you expect to test for during at-speed test.

But first, a little background. Test points were historically used in support of logic BIST by removing X-generators and making random resistant logic more testable. Logic BIST usage has grown significantly in recent years, mostly driven by the growth in silicon for automotive electronics and reliability requirements in the automotive and other missing critical industries market. Recently, EDT Test Points were introduced to reduce compressed ATPG pattern counts when using embedded ATPG compression. We’ve seen an average of 2x to 4x improvement in compression when EDT Test Points are added to a design. That means if you normally see 100x test pattern compression with ATPG alone, adding Test Points could get you to 200x to 400x compression, even with add at-speed tests—additional at-speed patterns can be added without sacrificing test time or cost such as transition patterns, cell-aware two-cycle patterns, and/or timing-aware patterns.

Back to the key question: will Test Points interfere with the ability of at-speed tests to detect defects?

The answer is clear that Test Points are safe for transition patterns that target gross delay defects. By definition, transition patterns use the transition fault model that targets a "gross" delay at every fault site. A transition pattern will detect faults by triggering a transition from a scan flop and capturing the result at a downstream scan flop. The fault model does not care about the path used. Furthermore, EDT Test Points use control points that are static (Figure 1). That means that they only help sensitize a path and will not shorten a functional path to be smaller than the function of the circuit. The one exception is if LOS (launch off shift) patterns are used. In such a case, an at-speed transition from the control test point can be launched as scan enable drops. This is easily prevented either by defining a false path from the test point or the test point can use a scan enable source signal and the functional logic use a pipelined scan enable.
By strict definition, EDT Test Points used during transition patterns will not interfere with the detection of transition faults.

The answer is more complicated for other at-speed test patterns. Unlike transition faults, timing-aware ATPG, used to detect small delay defects, considers the actual paths used during the tests. During timing-aware ATPG, the users can disable the EDT Test Points, which is simple to do. Timing-aware ATPG, however, will automatically find the longest paths around a fault site when creating patterns. Therefore, the addition of EDT Test Points will not necessarily hinder the ability to target and detect small delay defects.

EDT Test Points have enable signals, so more conservative users can disable them to ensure that functional paths are used and tested during at-speed test because Test Points can potentially shorten or interfere with the functional path. Even for conservative cases, though, it makes sense to allow control-type Test Points because they are only static signals that make the functional paths more easily sensitized. In such cases, the user has an option to have a separate control and observe Test Point activation when defining the test points.

There are also observe-type Test Points that can be added to improve the observability of specific nodes (Figure 2). During Test Points analysis, depending on the nature of the design, the number of control and observe test points are automatically optimized. The user, though, controls how many of each type and how many total number of test points will be inserted into a given physical region. Observe-type test points can make the functional path artificially short, so are often disabled during at-speed test.
Another concern I’ve heard has to do with the potential impact on functional timing if EDT Test Point logic is inserted on critical paths. There is an easy setup command in the ATPG compression tool to prevent the insertion EDT Test Points on critical paths. The same thing can be accomplished through test setup scripts.

There should also be no impact on path delay patterns since path delay targets specific functional paths that are defined by static timing analysis. Finally, when inserting EDT Test Points, false and multicycle paths can be ignored during testing by calling read_sdc or add_false_path commands during the insertion process.

Those are the basic considerations of using EDT Test Points with at-speed testing. Customers have a choice - without EDT Test Points, pattern sets can get too big for the tester and will need to be truncated, potentially lowering test quality. The trend, though, is to use EDT Test Points to improve pattern compression by 2-4x with static and at-speed tests. Then they can still target just the critical paths and generate a few additional at-speed patterns that have test points turned off. EDT Test Points result in a big improvement in compression and test time for all pattern types, along with some improvement in coverage too. The results are true for all test patterns.

Also see:
Design for Test Boot Camp, Part 1: Scan Test
Design for Test Boot Camp, Part 2: Test Compression
Design for test boot camp, Part 3: Advanced fault models and cell-aware test
Design for test boot camp, part 4: Built-in self test
Contactless testing
TAP interface gains popularity in board test
Design clock controllers for hierarchical test
New test points slash ATPG test pattern count
Launch-off-shift at-speed test
At-speed testing made easy