

Testing the Tester: Specification and Validation Approaches

John C. Johnson

Intel Corporation, Test Platform Architecture and Development

Specification Problems

It's a given that ATE test platforms are complex systems. Many HW and SW elements need to be integrated and work together for successful testing solutions. Too often we define a properly working platform within the context of the tools used to test and validate its behavior. Diagnostics, calibration, and special-purpose validation tools are used to prove a platform's adherence to specifications. However, these tools and usage conditions are usually far removed from the end-use applications environment. They ignore or bypass many of the interactions encountered by a user's actual test conditions. In an effort to manage the complexity of these interactions, some approaches try to limit the definition of a properly working tester to one that passes a particular validation standard (instead of meeting specifications in any legal usage).

Platform specifications should be valid in the context of end-use applications. The specifications should include a description of any usage limitations and remain valid within that envelope. Current attempts to tie the definition of timing accuracy to the results found by a 'standard' measurement tool are flawed. It is inevitable that end-use conditions will differ from a standard tool's environment. Allowing a validation result to define the spec leaves the user with no guarantee for that spec's performance in their applications, and is meaningless in the context of the application. Spec definition and validation methods are distinctly different and must not be merged to simplify the response to spec excursions.

Validation Tool Problems

It follows that validation tools also need to include comprehension and use of the application environment. Each user's test solution set will form a unique interface and resulting behavior within the tester's SW language, calibration tables, operating system configuration, etc. In Intel's experience, the most perplexing platform problems are of the "application fails, diagnostics pass", or tester-to-tester result mismatches types. Only application-like validation tools have a chance of resolving these issues.

System State Complexity

A growing challenge is the complex sequence of events and compatibility requirements between the many processes that establish "system state". It is common for a platform to require numerous initialization operations to reach a useful state. Boards may have NVRAM and PLD structures that are loaded with firmware versions in the factory and updated in the field. After power up, FPGA's receive configuration downloads. Then HW registers are placed in reset or initialization states. Calibration applies still more system

state values and these values may differ between calibration SW versions. The tester's runtime OS and user test program also establish specific states. How these sequences interact and maintain combinatorial compatibility is often the key to resolving intricate application problems.

A common example of a poorly architected diagnostics strategy is one where a forced sequence of hardware reset and tester calibration is required before diagnostics are able to run. This, of course, eliminates any possibility of detecting a system state corruption or calibration drift problem.

Approaches

Intel looks for adherence to very basic principles in the specification and validation of the ATE systems we procure:

1. Tester specifications need to be established within the context of application usage. For component VLSI ATE platforms, we look for device test programs running complex devices (either our own devices or devices of similar complexity). We also look for 'deviceless' application test programs to exercise a rich set of tests not possible with any single real device.
2. Spec validation needs to be performed in the same hardware and software state context as that encountered by a user's application. Timing accuracy tests should use the same system state, calibration paths, offset/des skew calculations, etc. as a user test program. Generally this means validation needs to be written as an application test program.
3. Validation tools need to be maintained and evolve for use in acceptance, regression, expert debug, and other testing scenarios. The validation test programs described in #1, for example, become very useful for SW release testing.
4. Diagnostics need to be capable of non-destructive testing of system setup and state content. We cannot be content with diags that only locate hard failures. These tools need to extend their capability to help locate and diagnose complex illegal, or out-of-spec system state failures.

Summary

The ATE industry should focus on both specification and validation tool standards - but specification guarantees should not be confused with the output results of a validation tool. Specifications and validation tools should be tied to end-use applications usage. Validation strategies need to comprehend and execute within the complex system state environment.