Test Flow Use Cases

Document Revision: January 7, 2003 9:15 am

Author: Jim O'Reilly

Informative statement

Priority ranking:

1. Must: Absolutely required for Minimum Viable Product
2. Want: Requested as an anticipated requirement

Use Case Overview:

1. Use Case #1: Dependent timing. Example of a test block with timing dependent upon search results from a search run from a test block in a prior flownode. For timing, often referred to as "source synchronous" - for instance, timing of a data strobe is dependent upon the position of a clock output.

1. Use Case #1: Dependent timing.

1.1. Priority - Must

1.1.1. Simplest straight through flow to be supported by P1450.4 extension

1.2. Assumptions/Prerequisites

1.2.1. This flow shows only the test program flow once all aspects of the program are loaded and tester requirements are initialized. If there are other flows to direct the loading of memories/registers and the initialization sequences, these must be discussed in an other Use Case.

1.2.2. No tester-to-DUT connect/disconnect sequences are encompassed in this flow

1.2.3. This sequence is only a subset of a complete flow.

1.2.4. This sequence is focused for single site testing

1.3. Pre-Conditions

1.3.1. All patterns are loaded into tester hardware. Timing, levels, and other test program have been loaded into the appropriate memory (either tester hardware, if the system stores this data in hardware, or as part of the run-time image of the test program). Tester is initialized.

1.4. Tasks/Scenario

1.4.1. A flow node which contains a search test is run. The timing or levels for this search depend only on values which are known at load time. The result of this search is stored so that a subsequent flow node can use the results.

1.4.2. A flow node which follows the first flow node runs a test method (functional test, dc test, timing or levels search, etc.) in which the input or output timing (or levels) depends on the results of the test method run in 1.4.1. Generally, the setup data for such a test will be represented as an equation of some sort; that equation will contain a variable for the dependent timing or voltage

1.5. Issues

1.5.1. One location in which the search result might be stored is in the "Meas" field of a Spec-Selector block. Alternatively, a Global variables block

could be defined (within the scope of a test flow or test program), and the result could be stored there.

1.5.2. It's quite likely that, even if the "Meas" field of a Spec-Selector block can be used, we'll also need a "Global variables" block of some sort - probably at the "test program" scope (I see the "test program" scope as encompassing the "test flow(s)" scope). As discussed in the previous conference call, we may have a completely user-defined set, or include a subset of the SEMI TSEM variables in addition to any user-defined variables.

1.5.3. In either case, the issue of persistance of the measured value arises. Once the result is obtained, is it valid for only the duration of the current test flow execution, or until reset? There are at least two cases to consider:

1.5.3.1. Valid only for the duration of the test flow. This case would apply to search results which will vary from device to device (i.e., the search result contains a device measurement). In such a case, we don't want to accidently use results from a previous device to calculate equations for the current device. So, some means of resetting the measured result (or flagging it as invalid) is required, either at the beginning or the end of a test flow execution.

1.5.3.2. Valid until the test program is reloaded. This case would apply to search results which do not vary from device to device. Examples would be (timing or levels) calibration of load board circuitry. Again, some means of resetting the measured result (or flagging it as invalid) is required; except in this case, the measured result is valid until the measurement is repeated, or the test program is reloaded.

1.5.4. When the result is used in a subsequent equation, that equation will contain a reference to the measured result. The syntax of such referencing needs to be worked out. From the 1450-1999 standard (p 73) there are a few examples of the usage of specs from a spec block to define the edge timing in a waveform table:

'txx' // simple variable from a spec sheet or labeled event
'txx*5' // expression
'txx+5ns' // expression

1.5.5. Generally, these specs will be resolved within the scope of whatever spec block is specified by the flow node (or by the PatternExec statement). We probably also need some syntax mechanism to specify a spec value from a spec block other than the one specified by the flow node (i.e., <SpecBlockName>.<specName>, perhaps).