RI Synapse: Automated Test Plan Optimization

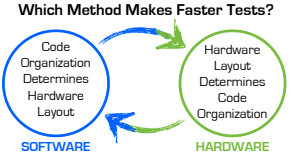
Making Measurements Fast

In production test, the three main objectives are: measurement accuracy, measurement consistency, and measurement speed. These are the criteria by which systems benchmark their performance, and speed is a crucial factor in the time to market and cost per part equation. The process of making measurements fast however, cannot be an afterthought in ATE design; it has to be integrated into the core of the test system to be effective. To address this need, RI developed Synapse: a sophisticated and powerful automated test plan optimizer. Combining Cassini's state-based software infrastructure and intelligent test code analysis, Synapse is able to fine-tune a user's test plan to execute faster and more efficiently in the hardware.

The Coding Conundrum

The difficulty of speeding up tests for a production environment is a two-fold problem:

- The user has to write efficient software code that reduces functions and tasks to their simplest form
- Compiled code has to be organized to execute efficiently for a given hardware setup on the tester



Careful programming can reduce wasteful and unnecessary processes, but this can become tedious and in some cases add significantly to the code development cycle. In addition, high-level program functions and calls reduce the amount of control a user can exert over the underlying processes and execution. The second and perhaps more difficult of the two is how to optimize the code for the hardware. For the most part, this undertaking lies outside the control of the user because it is difficult or impossible to manipulate compiled hardware code in a meaningful way. Secondly, the hardware setup can change from tester to tester, meaning what is optimal for one setup will not be optimal for another.

With Cassini this process is eliminated altogether by letting the user and the test system do what they do best. The user designs the measurements in a test plan using the software's hardware-aware function and command sets based on the attached instruments. Since the software is aware of the attached hardware, and knows how test code will be executed through state changes, the system has all the components it needs to begin optimizing. Synapse provides the logic engine for determining how to best organize the measurements to run fastest on the current hardware setup.

Conventional Test Programming & Compile

Test Plan Development

In a traditional test plan environment, the user writes commands and builds measurement functions using a text-based programming language. At this point, the algorithms and commands have no connection to how they will be executed in the ATE hardware.

Program Compile

Code compilation can be thought of as a two step process:

- The user's source code is translated into targeted-hardware code (colored blocks) by linking the software commands to their respective hardware resources.
- The targeted code is assembled to execute in the hardware exactly as it was organized by the user in software.

Executable Code

What comes out of this process is packaged code that can be used to drive the hardware. It has been verified against the capability of the system, and guaranteed to reflect the command and function structure the user has designed.

RI Test Plan & Compile

Test Plan Development & Compile

The RI test plans use test objects as the code building blocks, connected together in a graphical programming environment to represent functions and measurements. The test objects link to the resource capability of connected test instrument modules (TIMs). Connections made between blocks are compiled into sequences of state changes the hardware will execute to reflect the user's program.

Optimization

Synapse uses the hardware state information of the test objects to analyze the program execution behavior:

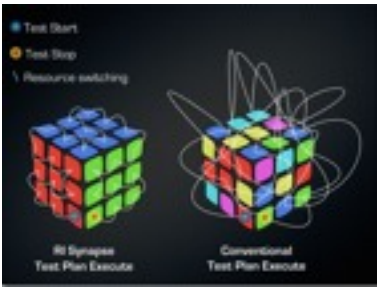
- Identify code that uses shared hardware resources (similar color), or redundant resource calls to combine or run tasks concurrently in the hardware.
- Synapse determines the time costs of hardware state switching, and optimizes the order measurements are executed (grouping colored blocks) to save time.

Executable Code

Using knowledge of the hardware and the user's program, Synapse has restructured the measurement organization to execute more efficiently on the tester. The result is packaged code that has been verified against the capability of the system, and guaranteed to reflect the command and function structure the user has designed.

Test Execution

When it comes time to execute the assembled code, it's easy to see the benefits of optimization. By organizing the order of measurements relative to the hardware, the system can execute the tests much more efficiently. The state changes between hardware switching have been greatly reduced, permitting much more effective use of the tester's time to gather and process data. Ultimately, automated optimization lets the tester solve the problem of speeding up tests and enables the test developer to focus on the specifics of accurately validating the part.



Differential Compiling

Often times a developer has limited access to the tester on a production floor, meaning less time to experiment with different test code. An added benefit of programming and optimizing state-aware coding blocks is the ability to exert more control over the compile stage. If just a section of the user's code has been altered, the system only needs to compile what has changed to incorporate the new information. This incremental compile ability gives the user another advantage in the test development cycle by accelerating the design iteration process.

Execution Order Control

Synapse was designed to aid the programmer, and make their job easier by finding the fastest order to execute tests. In some cases however, test setups and measurements need to be run on the part in a specific order. Because Synapse is integrated with the graphical test plan software, the user can view the order code will be executed, and control how and where optimization can be applied:

- Individual tests and measurement can be isolated from the optimizer
- Groups of sequential measurements can be excluded
- The optimizer can be completely disabled

By combining the abilities of automated test plan optimization with advanced software execution control, the user has a sophisticated and powerful production-test development environment in which to design.

Up Next

The next document in the series, "Test Management," describes the multi-functional software tool Guru. Combining test file and resource management, software version and distribution control, an automated backup system, and a networked communication backbone to tie it all together, Guru is the ultimate production test management client.