Verification Planning with Questa® Verification Management
by Kishan Kalavadiya and Bhavinkumar Rajubhai Patel, eInfochips

Verification of complex SoC (System on Chip) requires tracking of all low level data (i.e. regression results, functional and code coverage). Usually, verification engineers do this type of tracking manually or using some automation through scripting. Manual efforts in order to get above information while verifying complex SoC may lead us towards the delay in project execution. A verification planning tool can help to reduce such manual efforts and make the tracking process more efficient. Mentor, A Siemens Business has such a Verification Planning tool for QuestaSim within their Verification Management tool suite known as “Questa® Testplan Tracking”. This article contains detailed steps to use this tracking process along with key features which can reduce the time in verification cycle to track the verification progress.

INTRODUCTION
Mentor’s Verification Testplan Tracking facilitates the verification process management across all relevant aspects. Without such automation, tracking of verification progress requires a lot of manual effort in order to get information like coverage and regression status in a common place. Functional and code coverage numbers are key parameters in verification and without it verification cannot be closed. Regression analysis (individual test status) is also an important aspect of tracking the verification progress. Questa® Verification Management automates the tracking of verification progress by offering the analysis and optimization facilities followed by additional features like trend analysis, back annotation of results, filtration of specific data and other features. It provides the top level summary of all verification aspects, which provides real time visibility into the project to all parties, i.e. designers, verification engineers, project managers, etc. Due to this visibility, it becomes easy to manage risk factors and efficiency so project execution can be improved.

MOTIVATION FOR QUESTA® TESTPLAN DEVELOPMENT
Functional verification has been described as a major challenge in SoC designs with proper visibility into the verification process as a major factor contributing to this challenge. This lack of visibility impacts design quality, schedule predictability and cost.

A testplan is a document which captures the important features of a design and details regarding how they will be verified. Questa testplan is such a plan which can be linked directly to the coverage database and results can be annotated in that plan itself. By putting such a plan in place that captures list of verification intent, one can organize the tracking of verification progress better. While verifying the complex SoC (System on Chip), it may be possible that all the features we have listed in the Questa Testplan do not have equal priority. For that case, one can add separate user defined attributes in the testplan to provide priority for individual features. Verification engineers can focus their coverage closure efforts on the critical features that have been identified as a whole followed by important features and then by the “nice to have” features.

BASIC FLOW FOR QUESTA® VM VERIFICATION PLANNING
Questa Testplan includes feature wise test scenarios, functionality covered by that feature (functional coverage), assertions, code coverage and such more in XML format. After preparing such Questa Testplan, conversion of such plan in UCDB (Unified Coverage Database) format is required using xml2ucdb command. After running the regression, separate coverage databases will be available for individual test scenarios. At the time of merging of these coverage databases, Questa Testplan in UCDB format needs to be merged so that the results can be linked with the data available in Questa Testplan. One can
back annotate the results in XML format of Questa Testplan and the same results will be available in HTML report or in GUI as well. Creation of the Questa Testplan required certain steps and syntax, which is explained in the upcoming section.

**DETAILED STEPS FOR QUESTA® TESTPLAN DEVELOPMENT**

In many cases, the features will be verified in simulation and recorded as verified using coverage analysis. For Questa Testplan which facilitates back annotation of information, linking between the testplan and simulations is required. For that, certain document format must be followed. The format for Questa Testplan is described below with detailed steps.

**Step: 1. Install Questa add-in in Microsoft® Excel**

By default, Microsoft Excel does not contain Questa option in toolbar. So the user needs to install Questa add-in which is open source and freely available. After installing this add-in, it will be available in the toolbar as shown in figure 2.

**Step: 2. Prepare the Questa Testplan**

For creating the Questa Testplan, select create testplan option from Questa VM option as shown in figure 3.

Questa’s Testplan Tracking requires four distinct pieces of information for each requirement captured. They are **Section, Title, Link** and **Type**. **Link** is one of the attributes used while developing the Questa Testplan, which is required to link plan information to the actual results. The additional information for each requirement includes a **Description**, a **Weight** and a **Goal**. While these additional fields are not required, they are used the majority of the time. Questa’s Testplan Tracking also has the flexibility to allow user defined fields. If we look at the typically used fields in
a spreadsheet format, each field is represented by a column in the spreadsheet as shown in figure 4.

After preparing the Questa Testplan, it can be saved as XML or can be directly exported to XML from Questa VM options as shown in figure 3 on the previous page.

User can track different information by specifying the different options in Type field. Description for options available for Type field is as shown in table 1, below.

According to the requirement, user can set several rules as per the option available in Add Rule option. In figure 5, rule is applied to get the recursive toggle and statement coverage for some particular module, as shown on the opposite page.

**Step: 3. XML to UCDB Conversion of Questa Testplan**

Convert this XML Questa Testplan in UCDB using the command below:

```
xmll2ucdb -format Excel verification_plan.xml -ucdbfilename verification_plan.ucdb
```

### Table 1. Coverage Construct Type Field options

<table>
<thead>
<tr>
<th>Coverage Construct — in “Type” field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assertion</td>
<td>Assertion statement</td>
</tr>
<tr>
<td>Bin</td>
<td>Coverage item bin</td>
</tr>
<tr>
<td>Branch</td>
<td>Branch coverage scope</td>
</tr>
<tr>
<td>Condition</td>
<td>Condition coverage scope</td>
</tr>
<tr>
<td>CoverGroup</td>
<td>SystemVerilog covergroup statement</td>
</tr>
<tr>
<td>CoverPoint</td>
<td>SystemVerilog coverpoint statement</td>
</tr>
<tr>
<td>CoverItem</td>
<td>Generic name for any coverage or design object in a UCDB. This can be used to specify any objects not fitting into another category of construct.</td>
</tr>
<tr>
<td>Cross</td>
<td>SystemVerilog cross-coverage statement</td>
</tr>
<tr>
<td>Directive</td>
<td>PSL cover directives and SystemVerilog “cover” statements/properties</td>
</tr>
<tr>
<td>DU</td>
<td>All coverage on a given design unit</td>
</tr>
<tr>
<td>Expression</td>
<td>Expression coverage scope</td>
</tr>
<tr>
<td>Formal_Proff</td>
<td>Formal property of assertion object</td>
</tr>
<tr>
<td>Formal_Assumption</td>
<td></td>
</tr>
<tr>
<td>FSM</td>
<td>State Machine coverage scope</td>
</tr>
<tr>
<td>Instance</td>
<td>All coverage on a given instance</td>
</tr>
<tr>
<td>Rule</td>
<td>Forms a link using an automatically created virtual covergroup “User Rules” – either from a set of pre-defined Rules, or one you create.</td>
</tr>
<tr>
<td>Tag</td>
<td>Forms a link using any coverage tag command arguments which are specified in the Link column.</td>
</tr>
<tr>
<td>Test</td>
<td>Link to test attribute record. This is the test name.</td>
</tr>
<tr>
<td>Toggle</td>
<td>Toggle coverage scope</td>
</tr>
<tr>
<td>XML</td>
<td>Triggers hierarchical (nested) testplan import.</td>
</tr>
</tbody>
</table>
Step: 4. Regression and Coverage
Database generation
After preparing the Testplan, next step is to develop test cases to verify the features listed in the plan. After running the regression of those test cases, command for merge all separate coverage databases is as shown, above right:

```
vcover merge merged_cov.ucdb
<test1.ucdb> <test2.ucdb>…
<testn.ucdb> verification_plan.ucdb
```

We can back annotate the results in existing XML format Questa Testplan using option below:

```
Questa VM > coverage data > annotate
```

Step: 5. HTML
Report generation
Command for the HTML report generation from merged coverage database is as shown below:

```
vcover report -html -htmldir covreport
merged_cov.ucdb -details -testhitdata
```

VERIFICATION RESULTS ANALYSIS
Questa’s Testplan Tracking facilitates with multiple formats for analysis of the results hierarchically so that it will become easy to address the failures identified in regression and track the coverage holes to complete the verification process. Find below sample reports for analysis in different format.

Figure 6 and figure 7 contain the top level summary of Questa Testplan in GUI and HTML format respectively. Annotated Questa Testplan in XML format in figure 8 contains the same information which is shown in figure 6 and figure 7. Figure 8 (on the following page) shows the syntax for different Link options as per the requirements.

![Figure 5. Add Rule option in Questa Testplan](image)

![Figure 6. Verification Management Tracker in GUI](image)

![Figure 7. Questa Testplan in HTML](image)
Asterisk (*) is permissible in case of the same naming convention. Here in the above annotated Questa Testplan, section 2.5 containing 4 random test cases, out of which, 3 are passing and 1 is failing. To link these test cases with merged database, ‘*’ is being used in Link attribute. Here, pass/fail status of these test cases is in terms of coverage itself (i.e., getting 75% coverage for 4 objects means 3 test cases are passing out of 4 random cases). Similarly, we can get the results in cases of multiple iteration of the same test case.

Filtration is also one of the key features of Testplan Tracking, with which a user can filter out user-specific results from the whole report. A user can use Attribute Name, Coverage Numbers, Weight, Link Type, Instance Type and much more to filter the required data. Users need to create individual filters according to the requirements, as shown in the GUI illustrated in figure 9.

**TREND ANALYSIS**

Trend analysis offers the ability to track the progress of the coverage over time of all the objects. For trend analysis, user needs to manage one trend UCDB after each regression. Trend UCDB is nothing but a special purpose UCDB file which contains the coverage data over the period for trend analysis.

Trend reporting offers 2-dimensional representation of coverage number over the period for individual feature or instance of DUT. Trend report can be viewed in multiple formats (i.e. HTML Report, XML Report, etc.)

Trend reports appear in the GUI using trend UCDB at Verification Management > Browser > Trend Analysis

Command for trend UCDB generation from command line is as shown below:

```
vcover merge -trend [-output] <trend ucdb> <ucdb inputs>
```

Sample trend reports and a graphical representation for the same is shown in figure 10 and figure 11 respectively.
CONCLUSIONS
Verification of complex SoC projects is a complex process to manage without Verification Management. Automation of Verification Planning provides deeper visibility into the regression process to allow for throughput optimization. Questa VM’s Testplan Tracking Tool enables us to analyze the regression failures and coverage holes more efficiently by providing the reports in multiple formats. Use of Testplan Tracking can reduce the manual effort which is required to update the verification documents at the time of closure and quicken the documentation in a more appropriate format.

REFERENCES
Questa SIM Verification Management User Manual (Software Version 10.4c)
VERIFICATION ACADEMY
The Most Comprehensive Resource for Verification Training

30 Video Courses Available Covering
- UVM Debug
- Sequential Logic Equivalence Checking
- Portable Stimulus Basics
- SystemVerilog OOP
- Formal Verification
- Intelligent Testbench Automation
- Metrics in SoC Verification
- Verification Planning
- Introductory, Basic, and Advanced UVM
- Assertion-Based Verification
- FPGA Verification
- Testbench Acceleration
- PowerAware Verification
- Analog Mixed-Signal Verification

UVM and Coverage Online Methodology Cookbooks
Discussion Forum with more than 8100 topics
Verification Patterns Library

www.verificationacademy.com