Writing Verification Plan in Questa

ELECTGON
www.electgon.com
ma_ext@gmx.net

03.06.2019

Contents

Abstract

Test Plan for verification engineers is the guide or the metric which is used to measure results of coverage and other verification activities. Questa as a verification tool has its own procedure in order to enables verification engineers from checking if their verification plan is met or not. In this document we will show steps can be followed in order to use Questa to apply your verification plan.

1 What is a Verification Plan

Verification Plan is an organization of what should be touched in your simulation. It should include all your assertions, coverpoints, bins, crosses,.. so that a final report can be displayed showing what was achieved in your simulation and what is not achieved.

Verification Plan should be developed side by side with your verification environment. At the end we should map a relation between verification items (cover points, assertions ...) and test plan fields. This will be clear more in coming sections. What is important to know now is that verification items as it is constructed in your verification environement, it should be listed also in your verification plan so that verification tool (Questa in our documenet here) can be able to display test report at the end of your simulation.

This means also that these verification items should be listed in the plan with a predefined layout so that Questa can be able to recognize each verification item and be able to find it in verification environment. So in the following sections we will show this predefined layout of verification plan for Questa.

1.1 Procedures to Add Verification Plan in Questa

Mainly 4 steps are followed in order to successfully integrate a verification plan (or Test Plan) into your verification environment using Questa.

2 Step 1: Write Verification Plan

As mentioned before, there is special format for the Testplan in order to be sucessfully imported and understood by Questa. The easiest way is to write this plan in a spreadsheet orgainzed as discussed in the following subsection

2.1 Fields in the Verification Plan

The fields (or columns, in the case of a spreadsheet) of data in a testplan are determined positionally. In other words, for an import to work correctly, the fields to be imported must appear in their default (expected) positions in the plan.

The following Fields are mandatory fields that are required by Questa in order to be read successfully.

2.1.1 Section

Format: <section#>[.<subsection#>]...

- The section number, which is used to determine plan hierarchy and to generate tags (links to coverage items in the design).
- Duplicate Section numbers are not allowed and result in an error.
- You must define parent sections before child sections, for example: section 1 before 1.1, before 1.1.1, and so on.

2.1.2 Title

Format: <scope_name>

- The name of the test plan "section", where <scope_name> appears in the VM Tracker window and in the coverage reports.
- Use of special characters is discouraged ("*?/.").
- Duplicate Title entries (at a sibling level) are not allowed and result in an error.

An example of a title/name: if section 1 has a section title of "Interfaces", and section 1.1 has a section title of "Wishbone", then the hierarchical name of section 1.1 becomes /Interfaces/Wishbone.

2.1.3 Description

Format: <text>

Textual description of the Section. Usually, this would be a description of the testcase or test procedure that the section details. This can be free-form text.

2.1.4 Link

Format: <coverage_item> <coverage_item> ...

- Specifies the coverage objects or test data records which are linked to the testplan section.
- A testplan section can have one or multiple links to a coverage object or test data record.
- A specific format is required for the Link string itself.

2.1.5 Type

Format: type of <coverage_item> type of <coverage_item> ...

- Specifies the type of coverage item referred to in the Link field.
- There must be one entry in the "Type" column for each entry found in the "Link" column. However, there can be a single entry in the "Type" column which is applied to all the entries in the "Link" field.

2.1.6 Weight

Format: <weight_as_int>

An integer value used in the calculation of the weighted averages of the parent sections, calculated in floating point.

2.1.7 Goal

Format: <goal> (1 - 100)

- Specifies the percentage value as a goal for a particular coverage object in the GUI.
- This sets the threshold at which the bar-graph goes from uncovered (red) to covered (green) within the Questa SIM GUI.
- It does not affect the overall coverage grading calculation.

2.1.8 Path

Format: <path/to/linked/item>

- An optional data field, not included by default.
- Specifies the actual instance path to an item in a corresponding Link.
- Alternatively, an absolute path can be specified in the Link column (e.g. /top/a/cov1).

2.2 Summary of Verification Plan Fields

It is noted that all of these fields are free to enter any text according to user design except for the following fields that shall follow specified format:

- Section
- Link
- Type

Important:

- While adding 'Section' field you must write parent section before child section. And it should be numbers.
- While adding 'Type' field, inputs for this field are predefined and case sensitive. So it is allowed only to use one of the following values in this field

"Type" Field Entries		
Assertion	Cross	Instance
Bin	Directive	Rule
Branch	DU	Tag
Condition	Expression	Test
CoverGroup	Formal_Proof	Toggle
CoverPoint	Formal_Assumption	XML
CoverItem	FSM	

- While adding 'Link' field, for each type defined in previous table, the Link field has specific format. To make it simple, the Link field can accept full path of instance name (i.e. object that is needed to be tracked). In this case no need to write the path again in the Path field. Or it can accept only the instance name, i.e without the hierarichal path. In this case we have to add path to this instance.

Instance name then depends on the type of this instance (CoverPoint, Cross, Directive,...).

"Type" of Instance	Link Syntax
Assertion	Name (use Path field in this case) Full Path
Bin	Full Path
Branch	Full Path
Condition	Full Path
CoverGroup	Name (use Path field in this case) Full Path
CoverPoint	<CoverGroupType>::<CoverPointName>; <Full path to CoverGroupType>::<CoverPointName>;
CoverItem	Name (use Path field in this case) Full Path
Cross	<CoverGroupType>::<CrossName>; <Full path to CoverGroupType>::<CrossName>;
Directive	Name (use Path field in this case) Full Path
DU	Full Path
Expression	Full Path
Formal_Proof Formal_Assumption	Full Path
FSM	Full Path
Instance	Full Path

So it can be noticed that easiest way to remember this is to write full path to the object that you want to track. In case of Coverpoints and Cross, "::" is needed before coverpoint name.

2.3 Identifying Full Path of a Target

It may be ambiguous while Test Plan writing, what is the exact path of the object so that we can link it correctly in our Test Plan. The following steps may be helpful for identifying required path.

1. Save UCDB result of your simulation. Use the following command in your do file in order to save it:

```
coverage save -onexit <Target_Directory>/<filename.ucdb>
```

where <Target_Directory> is the directory you like to save the ucdb output in. <filename.ucdb> is your ucdb output.

2. Open your UCBD output by running the following command

```
vsim -viewcov <Target_Directory>/<filename.ucdb>
```

3. In Questa GUI you can find hierarichal structure of all your simulation objects as shown in figure 1.

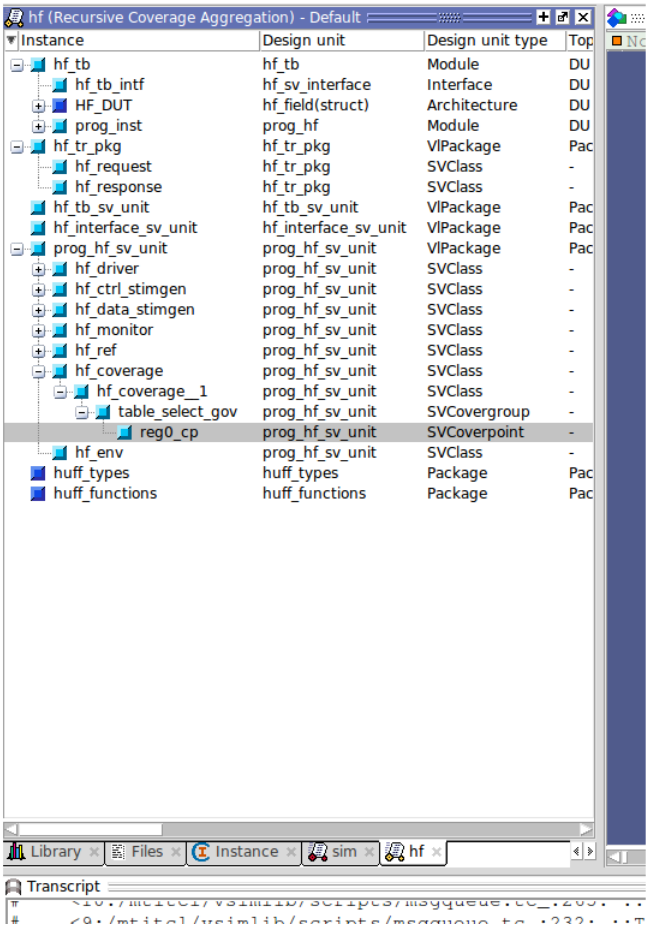
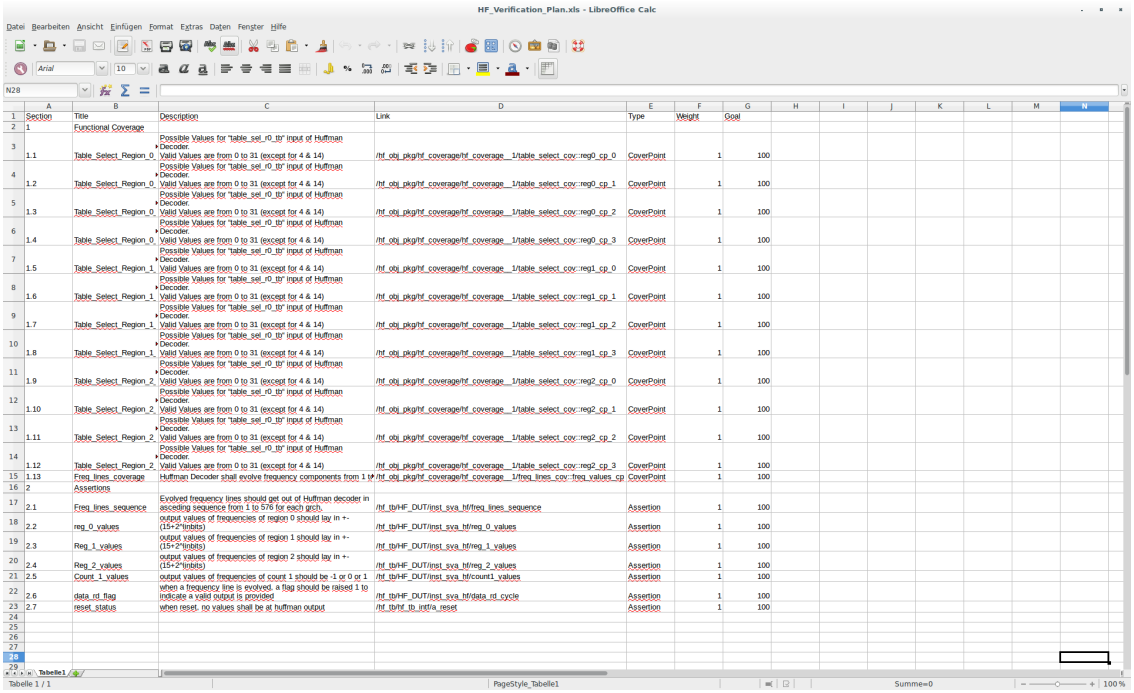


Figure 1: Objects Paths in Questa

Or as alternative, in Questa GUI top toolbar click on View>>Coverage>>Covergroup. In covergroup window you can check the path of your objects.

2.4 Testplan Sample

At the end, your Test plan should look like figure 2.



Section	Title	Description	Link	Type	Weight	Goal
1	Functional Coverage					
1.1	Table_Select_Region_0	Possible Values for "table_sel_r0_tb" input of Huffman Decoder. Valid Values are from 0 to 31 (except for 4 & 14)	/hf_obj_pkg/hf_coverage/hf_coverage__1/table_select_cov::reg0_cp_0	CoverPoint	1	100
1.2	Table_Select_Region_0	Possible Values for "table_sel_r0_tb" input of Huffman Decoder. Valid Values are from 0 to 31 (except for 4 & 14)	/hf_obj_pkg/hf_coverage/hf_coverage__1/table_select_cov::reg0_cp_1	CoverPoint	1	100
1.3	Table_Select_Region_0	Possible Values for "table_sel_r0_tb" input of Huffman Decoder. Valid Values are from 0 to 31 (except for 4 & 14)	/hf_obj_pkg/hf_coverage/hf_coverage__1/table_select_cov::reg0_cp_2	CoverPoint	1	100
1.4	Table_Select_Region_0	Possible Values for "table_sel_r0_tb" input of Huffman Decoder. Valid Values are from 0 to 31 (except for 4 & 14)	/hf_obj_pkg/hf_coverage/hf_coverage__1/table_select_cov::reg0_cp_3	CoverPoint	1	100
1.5	Table_Select_Region_1	Possible Values for "table_sel_r0_tb" input of Huffman Decoder. Valid Values are from 0 to 31 (except for 4 & 14)	/hf_obj_pkg/hf_coverage/hf_coverage__1/table_select_cov::reg1_cp_0	CoverPoint	1	100
1.6	Table_Select_Region_1	Possible Values for "table_sel_r0_tb" input of Huffman Decoder. Valid Values are from 0 to 31 (except for 4 & 14)	/hf_obj_pkg/hf_coverage/hf_coverage__1/table_select_cov::reg1_cp_1	CoverPoint	1	100
1.7	Table_Select_Region_1	Possible Values for "table_sel_r0_tb" input of Huffman Decoder. Valid Values are from 0 to 31 (except for 4 & 14)	/hf_obj_pkg/hf_coverage/hf_coverage__1/table_select_cov::reg1_cp_2	CoverPoint	1	100
1.8	Table_Select_Region_1	Possible Values for "table_sel_r0_tb" input of Huffman Decoder. Valid Values are from 0 to 31 (except for 4 & 14)	/hf_obj_pkg/hf_coverage/hf_coverage__1/table_select_cov::reg1_cp_3	CoverPoint	1	100
1.9	Table_Select_Region_2	Possible Values for "table_sel_r0_tb" input of Huffman Decoder. Valid Values are from 0 to 31 (except for 4 & 14)	/hf_obj_pkg/hf_coverage/hf_coverage__1/table_select_cov::reg2_cp_0	CoverPoint	1	100
1.10	Table_Select_Region_2	Possible Values for "table_sel_r0_tb" input of Huffman Decoder. Valid Values are from 0 to 31 (except for 4 & 14)	/hf_obj_pkg/hf_coverage/hf_coverage__1/table_select_cov::reg2_cp_1	CoverPoint	1	100
1.11	Table_Select_Region_2	Possible Values for "table_sel_r0_tb" input of Huffman Decoder. Valid Values are from 0 to 31 (except for 4 & 14)	/hf_obj_pkg/hf_coverage/hf_coverage__1/table_select_cov::reg2_cp_2	CoverPoint	1	100
1.12	Table_Select_Region_2	Possible Values for "table_sel_r0_tb" input of Huffman Decoder. Valid Values are from 0 to 31 (except for 4 & 14)	/hf_obj_pkg/hf_coverage/hf_coverage__1/table_select_cov::reg2_cp_3	CoverPoint	1	100
1.13	Freq_lines_coverage	Huffman Decoder shall evolve frequency components from 1 to 576	/hf_obj_pkg/hf_coverage/hf_coverage__1/freq_lines_cov::freq_values_cp	CoverPoint	1	100
2	Assertions					
2.1	Freq_lines_sequence	Evolved frequency lines should get out of Huffman decoder in ascending sequence from 1 to 576 for each grch.	/hf_tb/HF_DUT/inst_sva_hf/freq_lines_sequence	Assertion	1	100
2.2	reg_0_values	output values of frequencies of region 0 should lay in +-(15+2^lnbits)	/hf_tb/HF_DUT/inst_sva_hf/reg_0_values	Assertion	1	100
2.3	Reg_1_values	output values of frequencies of region 1 should lay in +-(15+2^lnbits)	/hf_tb/HF_DUT/inst_sva_hf/reg_1_values	Assertion	1	100
2.4	Reg_2_values	output values of frequencies of region 2 should lay in +-(15+2^lnbits)	/hf_tb/HF_DUT/inst_sva_hf/reg_2_values	Assertion	1	100
2.5	Count_1_values	output values of frequencies of count 1 should be -1 or 0 or 1	/hf_tb/HF_DUT/inst_sva_hf/count1_values	Assertion	1	100
2.6	data_rd_flag	when a frequency line is evolved, a flag should be raised 1 to indicate a valid output is provided	/hf_tb/HF_DUT/inst_sva_hf/data_rd_cycle	Assertion	1	100
2.7	reset_status	when reset, no values shall be at huffman output	/hf_tb/hf_tb_intf/a_reset	Assertion	1	100

Figure 2: Testplan Sample

3 Step 2: Export the Plan to XML

This can be done by saving your spreadsheet as microsoft Excel 2003 XML.

4 Step 3: Import XML plan into Verification Environment

Use the following command to import the xml plan

```
xml2ucdb -format Excel xml_plan_name.xml <Target_Dir>/<imported_name.ucdb>
```

5 Step 4: Merge Imported Plan with Simulation Data

Use the following command to merge your plan with simulation results

```
vcover merge -totals <Target_Dir>/<Merged_plan.ucdb> <imported_name.ucdb> <
    Test1_result.ucdb> <Test2_result.ucdb> <Test3_result.ucdb> ...
```

note that option -totals is used to include all coverage items (assertions, coverpoints, cross,..) into your test tracing.

Bibliography

[1] Questa SIM Verification Management User's Guide.

[2] Questa SIM Command Reference Manual.

[3] Lecture Notes/Script, Prof. Dr. Ing. Stefan Wolter – Hochschule Bremen – Spring 2015.