OSVVM based Scripting Approach

The intent of this approach is to separate the complicated scripts that are needed to run the tools from the scripts that actually describe the structure of a design and what to do with it. This will allow people working on a project to focus on what needs to be done rather than worrying about the steps to get a particular tool to do it.

Going further, when a simulation model that is compliant with this approach is added, the only thing that needs to be done is tell the scripts where the directory containing the files is.

This is an evolving approach. This is the second revision and while it has changed significantly from the first revision, it still supports the style of scripting supported in the first revision. Input is welcome.

One notable change is that all basic named scripts are lower case.

1. Simplified Approach

To build the entire project there are three steps:

1) In the simulator, go to directory simulation/scripts
2) Run the script, "do StartUp.tcl"
3) Build the design: "build"

2. Project Files

A project file allows the specification of basic tasks to run a simulation: library, analyze, include, and simulate. The naming of the project file is of the form <Name>.pro.

An example of a project file is:

```
include ../Models
library Lib_Model1
analyze file1.vhd
simulate fred
```

The items include, library, analyze, and simulate are TCL procedures. Hence, a project file is actually a TCL file, and if necessary, TCL can be used, however, the intent is to keep it simple.

3. Commands

include <directory> **include <path>/<file>**	Include accepts either a file or a directory. If it is a file and its extension is .tcl, .do, or .pro, the file will be sourced. If it is a file and its extension is .files or .dirs it will be handled in a manner consistent with revision 1 of the scripts. If it is a directory, then files whose base name matches the directory name and that have the extensions .pro, dirs, files, .tcl, and .do are searched for (in this order) and processed as above if they exist.
library <library>	Make the library the active library. If the library does not exist, create it and create a mapping to it.
analyze <file>	Compile the file.
simulate <design unit>	Start a simulation on the design unit. Often best if this is a configuration name.
build <directory> **build <path>/<file>**	Re-initializes the working directory to the script directory, opens a transcript file, and calls include.
map <library> [<path>]	Create a mapping to a library
RemoveAllLibraries	Delete all of the working libraries.
MapLibraries	Create a mapping to libraries

4. Running simulations

The commands can be used while running the simulator interactively. Hence, one can activate the library LIB_TEST, compile the testbench architecture TestCtrl_Test1.vhd, and simulate the configuration Test1 by typing the following in the simulator:

```
library LIB_TEST
analyze ../Tests/TestCtrl_Test1.vhd
simulate Test1
```

5. About the scripts

StartUp.tcl	Sets up locations for scripts, libraries, and logs. It calls the other scripts to set up the entire environment.
ToolConfiguration.tcl	Settings to detect and configure a tool to run within the scripts. Provides options and settings.
OsvvmProjectScripts.tcl	Create the procedures to run the simulations.