OSVVM based Scripting Approach

The OSVVM scripting approach uses descriptor files to inform the scripts where the files are located (directories), what libraries to use (or create), what files to compile, and what simulations to run. The intent is to allow the people working on the project to solely focus on what needs to be done rather than being worried about how it is done.

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. So it may change in the future. Input is welcome.

1. Initialization

The scripts need to be initialized. You can do this each time you start a simulation by doing:

```
source StartUp.tcl
```

Alternately in Aldec RivieraPro, set the environment variable, ALDEC_STARTUPTCL to StartUp.tcl (including the path information). Similarly in Mentor tools, set the environment variable, MODELSIM_TCL to StartUp.tcl (including the path information).

Simulations can be run out of the Scripts directory, but it is recommended to run the simulations out of a separate directory dedicated to this purpose (such as sim). This means cleanup is just a matter of deleting the sim directory and recreating a new one.

2. Simplified Approach

To run a project, it is simply a matter of running the build script. The build script can include path names. If you checkout the OSVVM library and the VerificationIP library, into the same directory, the directory hierarchy will be:

- osvvm – OSVVM library release from GitHub
- VerificationIP – OSVVM Verification IP library from GitHub
    - AXI4 - git submodule with AXI4 verification components from GitHub
        - Axi4Lite
            - src – Axi4Lite master, slave, and monitor (to be released)
            - testbench

          AxiStream – AixStream master slave
            - src – AxiStream master, slave models
            - testbench
        - common – shared packages
    - Scripts – git submodule with OSVVM scripting library from GitHub
    - sim – created by you as part of the steps below

Create a directory named sim at the same level as the Scripts directory (see design hierarchy above). In the simulator change directory into sim. You can compile the OSVVM and OSVVM_AXI4 libraries, and run the AXI4 Lite and AXI4 Stream simulations by doing:

```
build ../../osvvm
build ../AXI4/RunTests.pro
```

3. 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. The tasks are TCL procedures. Hence, a project file is actually a TCL file, and when necessary, TCL can be used, however, the intent is to keep it simple.

An example of a project file is:

```
include  ./Axi4.pro
library  osvvm_TbAxi4Lite
analyze  TestCtrl_e.vhd
analyze  TbAxi4Lite.vhd
analyze  TbAxi4Lite_BasicReadWrite.vhd
simulate TbAxi4Lite_BasicReadWrite
```

Path names specified in a project file are relative to the location of the project file.

4. Commands

include <directory> include <path>/<file>	Include accepts either a file or a directory. If it is a file and its extension is .pro, .tcl, or .do, 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 any found will be processed as above if they exist. A path name specified is relative to the location of the current <file>.pro directory location.
library <library>	Make the library the active library. If the library does not exist, create it and create a mapping to it. Libraries are created either in the path specified by LIB_BASE_DIR in Scripts/StartUp.tcl or the directory from which the first build is run.

analyze <file>	Compile the file. A path name specified is relative to the location of the current <file>.pro directory location. Library is the one specified in the previous library command.
simulate <design unit>	Start a simulation on the design unit. Often best if this is a configuration name. Library is the one specified in the previous library command.
build <directory> build <path>/<file>	Re-initializes the working directory to the script directory, opens a transcript file, and calls include. A path name specified is relative to the location of the current <file>.pro directory location.
map <library> [<path>]	Create a mapping to a library
RemoveAllLibraries	Delete all of the working libraries.
MapLibraries	Create a mapping to libraries

5. 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
```

6. 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.

7. Older Descriptor Files

It is intended that the older descriptor files are deprecated. Each directory may have three types of descriptor files: directories, files, and simulations.

A directory descriptor file is of the form <DirectoryName>.dirs and contains a list of directories to search for other descriptor files.

A file descriptor file is of the form <DirectoryName>.files and contains a list of names that are either .vhd for vhdl, .v for Verilog or .lib for current working library. If the file descriptor file contains a .lib, it sets the current working library to that name without the .lib extension and that library will be used until it is changed or the end of file is reached.

A simulation descriptor file is of the form <DirectoryName>.sim. If a name contains .lib, it specifies the current working library to run simulations from. If a names contains .vhd, it is compiled. If a name does not contain an extension, it is assumed to be a library unit for simulation. Currently any text following the library unit is passed to the simulator – this may be replaced by mechanism to run a script name.