LHC Research Ubuntu/ModelSim Tutorial

Lev Kurilenko (levkur@uw.edu)
Anthony Faubert (deadbeef@uw.edu)
4/15/18

Table of Contents

Introduction

This tutorial covers the more advanced aspects of ModelSim that people in Dr. Hauck's ACME group at the University of Washington should be familiar with. The tutorial assumes a basic understanding of ModelSim and how to go about simulating a design using a .do file. This particular tutorial is for Ubuntu/Linux users and it contains notes for Vivado on Linux in addition to ModelSim.

Vivado USB drivers for Ubuntu

For Linux, there's an extra step to setting up Vivado that wasn't covered in the Vivado tutorial document. Vivado requires USB drivers to be installed for the KC705 board. These drivers come with Vivado, but require superuser permissions to install and so are not installed when Vivado is installed. Assuming Vivado 2017.4 was installed in /usr/local/xilinx, execute the following commands to install the drivers:

```
cd /usr/local/xilinx/Vivado/2017.4/data/xicom/cable_drivers/lin64/install_script/install_drivers
sudo ./install_drivers
```

For more detailed instructions on installing the USB drivers, see the following document:
https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_4/ug973-vivado-release-notes-install-license.pdf

ModelSim Installation

Due to the nature of some of the projects, an unlicensed and slow version of ModelSim can sometimes be inadequate. Mentor Graphics deliberately slows down the program to motivate customers to purchase a license and increase the speed of their simulations. University of Washington provides a faster version of ModelSim which uses a floating license.

The download link can be found here:

https://downloads.engr.washington.edu/downloads/distribution/Mentor_Graphics/SE%2010.5c/

Download the following files:

install.linux64

modelsim-base.mis
modelsim-docs.mis
modelsim-linux_x86_64.mis
modelsim-gcc-linux_x86_64.mis

If you have “Linux Redhat 5 and greater” or “Suse 10 or greater.” then you might need to download modelsim-linux.mis as well. Download the INSTALL_NOTES.txt file and CTRL-F “For Unix or Linux” for details.

Once the download completes, ensure all the *.mis files are in one directory, run “install.linux64” and go through the installation process. Under "select source" choose the folder where the *.mis files are. Make sure to mark all 4 components to be installed. I recommend that you install it to /usr/local/modelsim

Setting up the License/Running ModelSim

The license provided by UW is a floating license. To set up the license, the environment variable “LM_LICENSE_FILE” must be set to “ 27007@mentorls.s.uw.edu ” before running ModelSim. I personally prefer to write a script to launch ModelSim:

```
#!/bin/bash
INSTALL_LOC=/usr/local/modelsim
export LM_LICENSE_FILE=27007@mentorls.s.uw.edu
$INSTALL_LOC/modeltech/bin/vsim "$@"
```

`INSTALL_LOC` should be changed to the directory in which you installed ModelSim.
At this point you should run ModelSim (using that script or your own means) and make sure everything is working. If it fails to run due to missing libraries, run `sudo apt-get install libxft2 libxft2:i386 lib32ncurses5` if you’re running Ubuntu. If you are running a different distro, you’ll have to figure out the equivalent packages.

Compiling Vivado Simulation Libraries

Oftentimes, IP cores provided by the vendor are used in a digital design. The IP cores contain HDL code, and sometimes hardened IP blocks, that target specific functionality. When an IP core is included in a design, it is important to still have the ability to simulate the design. To accommodate for this, vendors usually provide a functional simulation model of the IP core. This allows the user to understand the functionality of the IP core and incorporate it into the simulation flow.

The functional simulation model uses libraries created by the vendor specifically for simulation. Examples of Xilinx libraries include ‘unisim’, ‘secureip’, and ‘unimacro’. These libraries need to be included as a ModelSim library to simulate any design containing an IP core that references these libraries. Described below are the steps for compiling Vivado simulation libraries and including them into your ModelSim simulation.

1. Open the 2017.4 Vivado IDE.

2. In the top, select the ‘Tools’ tab and select ‘Compile Simulation Libraries…’ as shown in Figure 5.

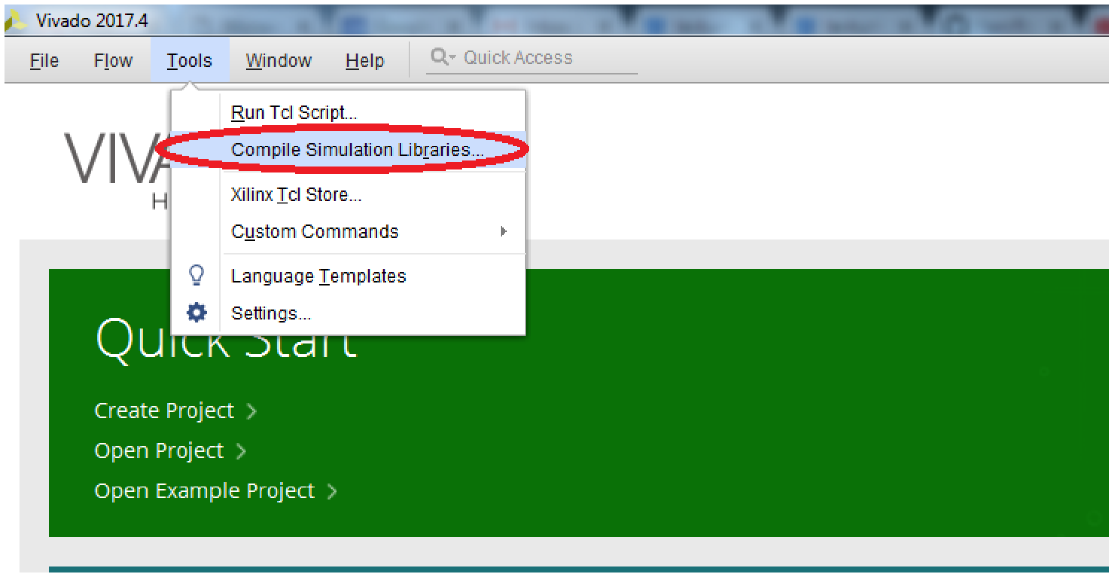
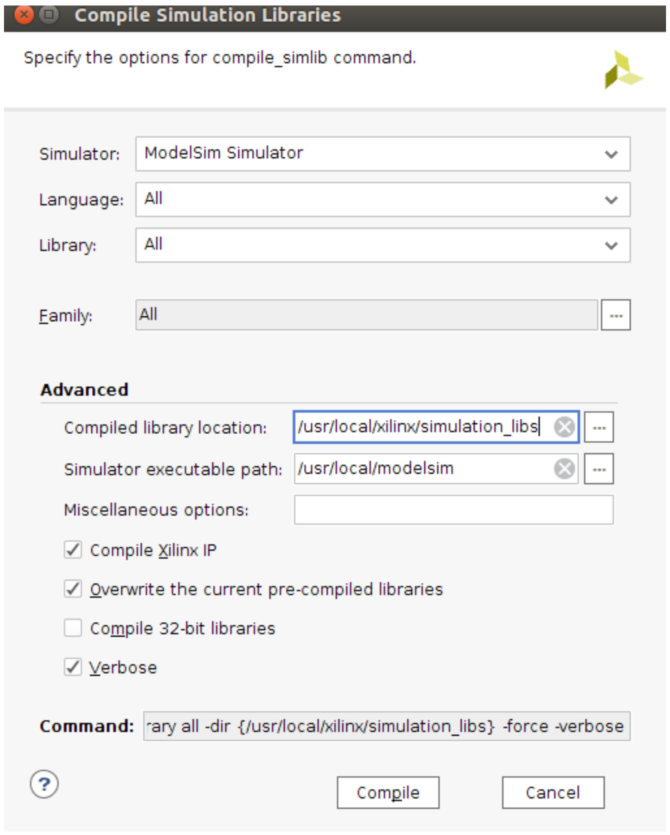


Figure 5: Select compile simulation libraries

3. A ‘Compile Simulation Libraries’ window will open as shown in Figure 6. Make sure all the options shown are selected and proper paths specified, then hit Compile.



Compile Simulation Libraries

Specify the options for compile_simlib command.

Simulator: ModelSim Simulator

Language: All

Library: All

Family: All

Advanced

Compiled library location: /usr/local/xilinx/simulation_libs

Simulator executable path: /usr/local/modelsim

Miscellaneous options:

[✓] Compile Xilinx IP

[✓] Overwrite the current pre-compiled libraries

[ ] Compile 32-bit libraries

[✓] Verbose

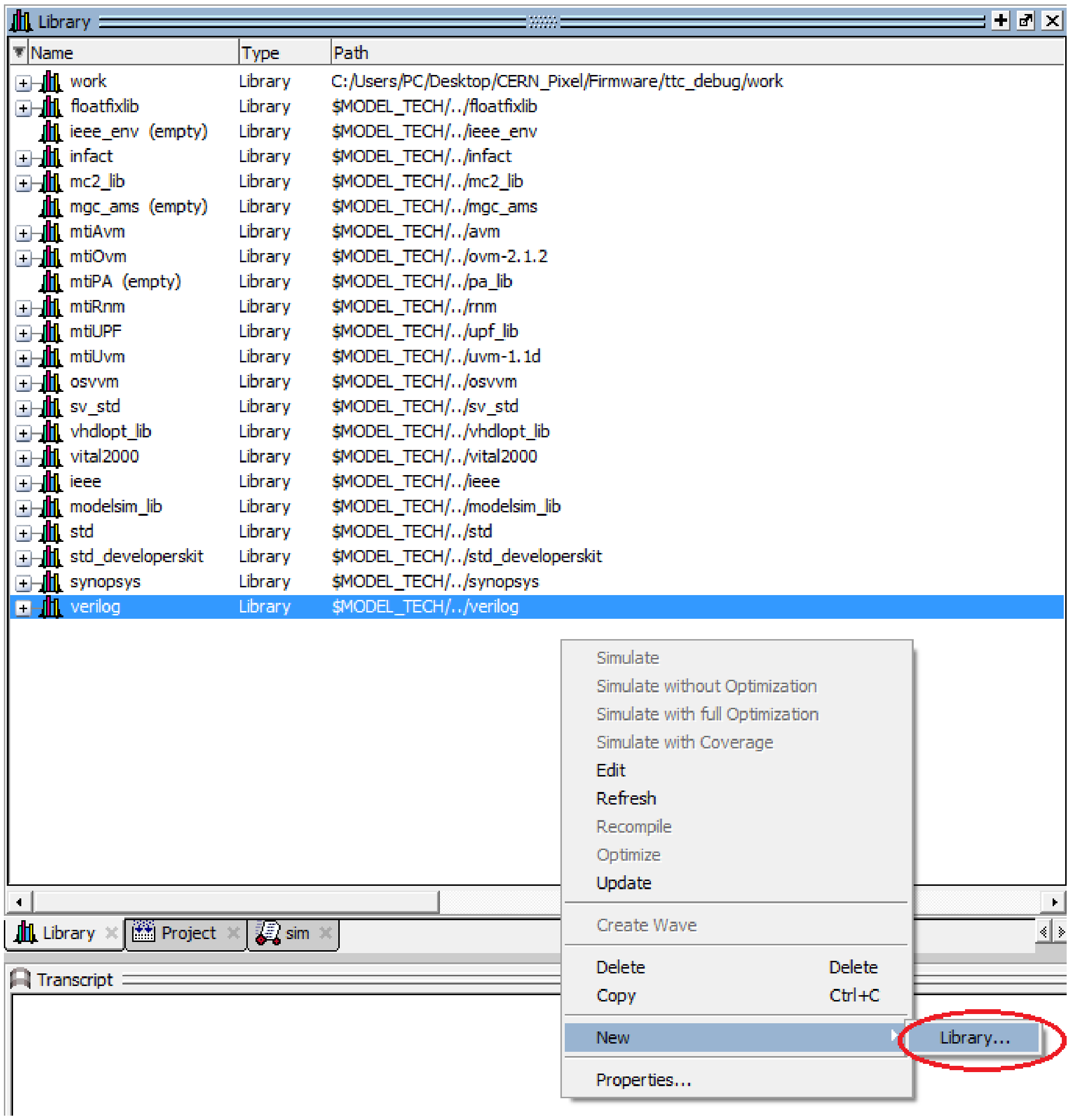
Command: `rary all -dir {/usr/local/xilinx/simulation_libs} -force -verbose`

Figure 6: Compile simulation libraries window

4. Compiling the libraries may take several minutes.

5. Once the libraries are compiled, any library used by an IP core must be included as a ModelSim library.

6. Right-click in the ModelSim Library pane, hover the mouse over ‘New’, and select ‘Library…’ as shown in Figure 7.



Name	Type	Path
work	Library	C:/Users/PC/Desktop/CERN_Pixel/Firmware/ttc_debug/work
floatfixlib	Library	$MODEL_TECH/../floatfixlib
ieee_env (empty)	Library	$MODEL_TECH/../ieee_env
infact	Library	$MODEL_TECH/../infact
mc2_lib	Library	$MODEL_TECH/../mc2_lib
mgc_ams (empty)	Library	$MODEL_TECH/../mgc_ams
mtiAvm	Library	$MODEL_TECH/../avm
mtiOvm	Library	$MODEL_TECH/../ovm-2.1.2
mtiPA (empty)	Library	$MODEL_TECH/../pa_lib
mtiRnm	Library	$MODEL_TECH/../rnm
mtiUPF	Library	$MODEL_TECH/../upf_lib
mtiUvm	Library	$MODEL_TECH/../uvm-1.1d
osvvm	Library	$MODEL_TECH/../osvvm
sv_std	Library	$MODEL_TECH/../sv_std
vhdlopt_lib	Library	$MODEL_TECH/../vhdlopt_lib
vital2000	Library	$MODEL_TECH/../vital2000
ieee	Library	$MODEL_TECH/../ieee
modelsim_lib	Library	$MODEL_TECH/../modelsim_lib
std	Library	$MODEL_TECH/../std
std_developerskit	Library	$MODEL_TECH/../std_developerskit
synopsys	Library	$MODEL_TECH/../synopsys
verilog	Library	$MODEL_TECH/../verilog

Figure 7: Add Library to ModelSim

7. In the ‘Create a New Library’ window select the ‘a map to an existing library’ option. Enter the library name of the library you wish to create. Press ‘Browse’ in the ‘Library Maps to:’ section as shown in Figure 8.

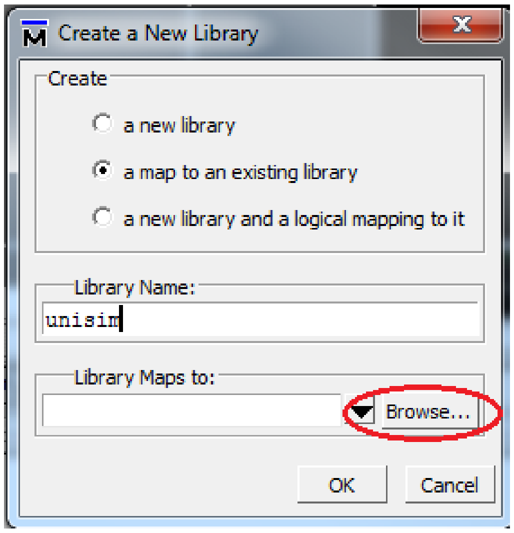


Figure 8: Create a New Library

8. Navigate to the simulation_libs folder where the Xilinx simulation libraries were compiled. In there, find the library corresponding to the library name specified in step 7. See figure 9.

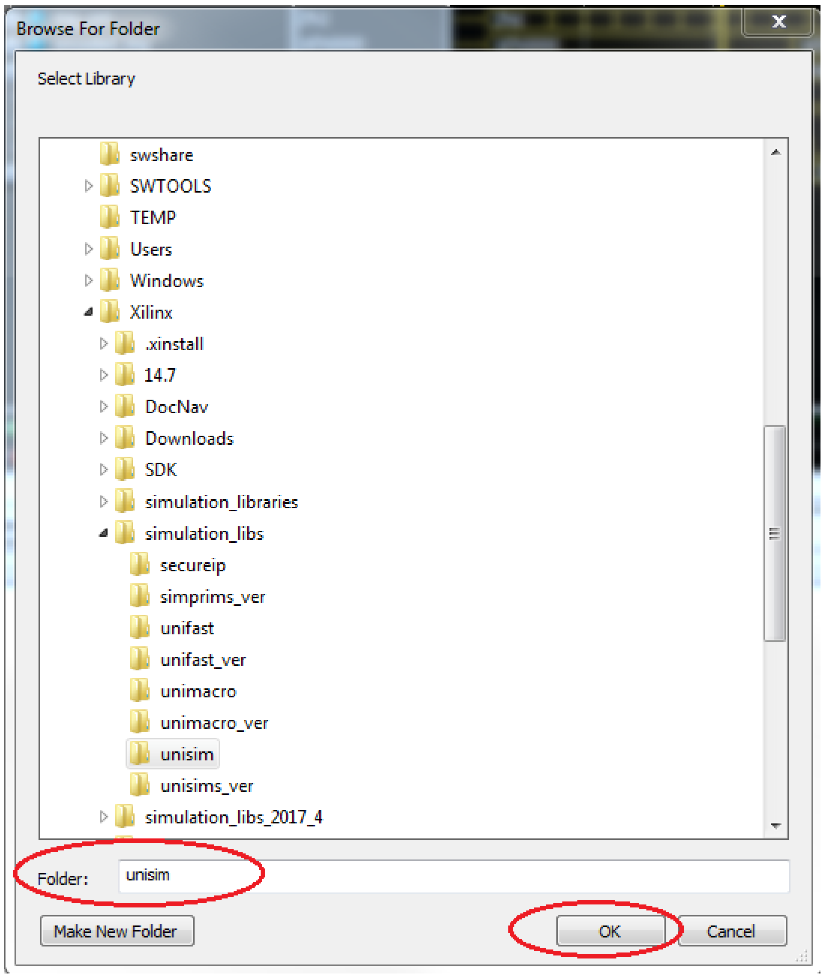
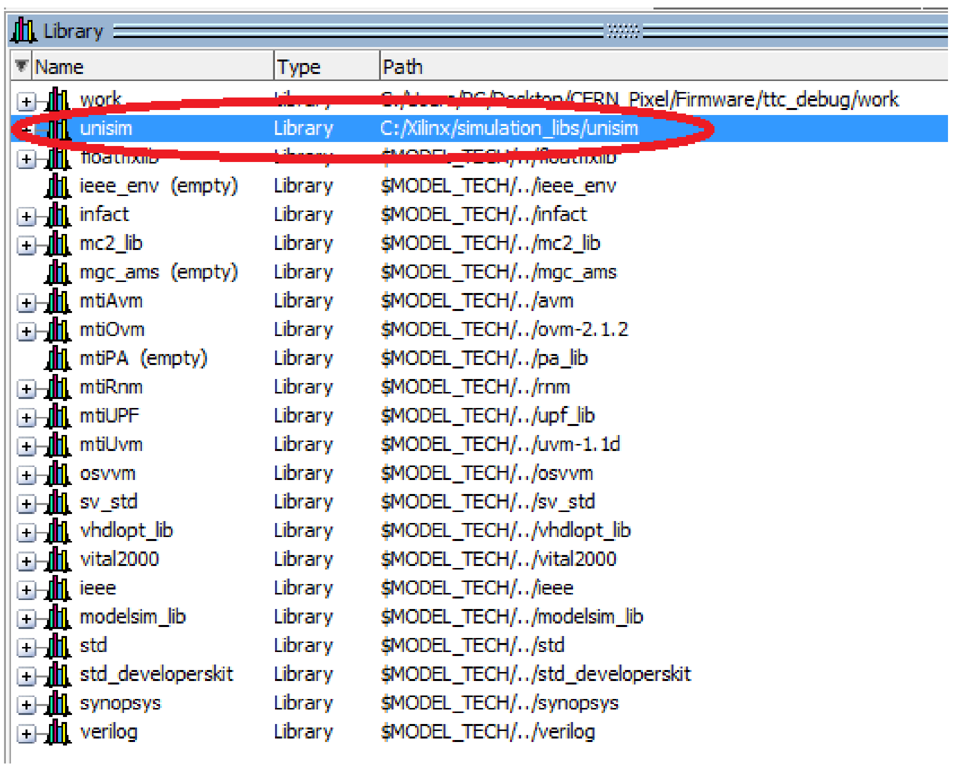


Figure 9: Select Library folder corresponding to Library Name

9. The library is now included in the library pane as shown in Figure 10.



Library

Name	Type	Path
work	Library	C:/Users/PC/Desktop/CERN_Pixel/Firmware/ttc_debug/work
unisim	Library	C:/Xilinx/simulation_libs/unisim
floatfixlib	Library	$MODEL_TECH/../floatfixlib
ieee_env (empty)	Library	$MODEL_TECH/../ieee_env
infact	Library	$MODEL_TECH/../infact
mc2_lib	Library	$MODEL_TECH/../mc2_lib
mgc_ams (empty)	Library	$MODEL_TECH/../mgc_ams
mtiAvm	Library	$MODEL_TECH/../avm
mtiOvm	Library	$MODEL_TECH/../ovm-2.1.2
mtiPA (empty)	Library	$MODEL_TECH/../pa_lib
mtiRnm	Library	$MODEL_TECH/../rnm
mtiUPF	Library	$MODEL_TECH/../upf_lib
mtiUvm	Library	$MODEL_TECH/../uvm-1.1d
osvvm	Library	$MODEL_TECH/../osvvm
sv_std	Library	$MODEL_TECH/../sv_std
vhdlopt_lib	Library	$MODEL_TECH/../vhdlopt_lib
vital2000	Library	$MODEL_TECH/../vital2000
ieee	Library	$MODEL_TECH/../ieee
modelsim_lib	Library	$MODEL_TECH/../modelsim_lib
std	Library	$MODEL_TECH/../std
std_developerskit	Library	$MODEL_TECH/../std_developerskit
synopsys	Library	$MODEL_TECH/../synopsys
verilog	Library	$MODEL_TECH/../verilog

Figure 10: Xilinx simulation library shows up in ModelSim Library pane

After these steps are completed, you are ready to simulate designs containing functional simulations of Xilinx IP blocks.

Simulation

Simulations can be created in the ModelSim Transcript window using tcl commands. The simulation creation process can be automated by creating a .do file containing tcl commands and executing the .do file by running:

```
do filename.do
```

in the Transcript window. A detailed tutorial can be found in Dr. Hauck's EE271 tutorial on pages 6-14:

https://class.ee.washington.edu/271/hauck2/labs/Quartus%20Tutorial.pdf