Simulation Automation Scripts

Roughly 2/3 of the simulation automation scripts are written in TCL (.do), and Makefile takes about the majority of the another part, while Python, Perl, bash, batch file etc are only minimal used. Therefore we will use TCL (do) as the automation scripting language.

do File

QuestaSim

QuestaSim using the do command.

A .do file is passed to QuestaSim over the command line, it is a .tcl script that tells QuestaSim what to do.

Writing batch files for simulation in Modelsim/QuestaSim

https://electronics.stackexchange.com/questions/83952/writing-batch-files-for-simulation-in-modelsim-questasim

https://www.programmersought.com/article/42061382137/
https://www.programmersought.com/article/20904718176/
https://avi-brown.medium.com/how-to-use-do-files-in-modelsim-vhdl-simulations-dd8e5a663a5a
https://www.reddit.com/r/FPGA/comments/nojuri/questasim_dumb_question_how_to_make_a_makefile_or/?rdt=60808
https://community.intel.com/t5/Intel-Quartus-Prime-Software/QUESTA-do-file/m-p/1528943

QuestaSim -- Dumb Question: how to make a makefile or .do file?
https://www.reddit.com/r/FPGA/comments/nojuri/questasim_dumb_question_how_to_make_a_makefile_or/

To perform a functional simulation with the QuestaSim software with command-line commands
https://www.intel.com/content/www/us/en/programmable/quartushelp/17.0/mapIdTopics/jka1465596899648.htm

How to use .do files in ModelSim VHDL simulations
https://avi-brown.medium.com/how-to-use-do-files-in-modelsim-vhdl-simulations-dd8e5a663a5a

ModelSim

Using DO files in ModelSim

ModelSim Compile Script
https://www.doulos.com/knowhow/tcltk/example-tcl-and-tcltk-scripts-for-eda/modelsim-compile-script/

How to automatically simulate the top-level VHDL entity with ModelSim?
https://stackoverflow.com/questions/20589492/how-to-automatically-simulate-the-top-level-vhdl-entity-with-modelsim/

modelsim script for compile all

https://stackoverflow.com/questions/49538638/modelsim-script-for-compile-all

Vivado

Vivado Simulator scripted flow Part 1: Basic CLI usage
https://www.itsembedded.com/dhd/vivado_sim_1/
https://www.itsembedded.com/dhd/vivado_sim_1/

Vivado also allows the user to perform the design flow using TCL language.
https://surf-vhdl.com/tcl-script-vivado-project-tutorial/

Vivado Tcl Build Script
https://projectf.io/posts/vivado-tcl-build-script/

Using Tcl Commands and Scripts
https://docs.amd.com/r/en-US/ug900-vivado-logic-simulation/Using-Tcl-Commands-and-Scripts

Vivado non-project mode TCL script for simulation? (SystemVerilog)
https://www.reddit.com/r/FPGA/comments/i36o4x/vivado_nonproject_mode_tcl_script_for_simulation/?rdt=62130

Using Vivado in tcl mode.
https://www.eevblog.com/forum/fpga/using-vivado-in-tcl-mode/

Automate the usage of vivado gui by using tcl scripts
https://stackoverflow.com/questions/55495977/automate-the-usage-of-vivado-gui-by-using-tcl-scripts

How TCL Can Interact with SystemVerilog / UVM?

TCL to manipulate systemverilog / uvm

TCL (Tool Command Language) interacts with SystemVerilog UVM (Universal Verification Methodology) environments, specifically within the context of hardware verification simulations, in a few key ways:

1. Simulator Interaction:

TCL as a simulator shell: Many SystemVerilog simulators provide a TCL prompt or interface that allows users to interact with the simulation. You can use TCL commands to control the simulation flow, such as:

- Starting and stopping the simulation.
- Setting breakpoints.
- Stepping through the simulation.
- Querying the values of signals and variables.
- Calling SystemVerilog functions or tasks (possibly through an intermediary C interface).

Running TCL scripts: You can create TCL scripts to automate simulation tasks, such as running a series of tests, generating reports, or manipulating simulation settings.

2. Debugging and Control:

Interactive debugging: The simulator's TCL prompt can be used in conjunction with UVM libraries to create an interactive debugging environment. For example, some UVM libraries provide functions that, when called from SystemVerilog, can pause the simulation and allow you to interact with it through the TCL prompt, enabling you to inspect UVM objects and potentially alter the simulation flow.

Driving test cases: TCL can be used to drive test cases and provide input to the UVM testbench. This allows you to control the simulation from a scripting level, which can be useful for test automation and regression testing.

3. Indirect Interaction through DPI (Direct Programming Interface) and C:

TCL to C to SystemVerilog: TCL does not have a direct DPI like C. However, you can use C as an intermediary layer. You can write C code that interacts with SystemVerilog through DPI and then call that C code from your TCL scripts. This allows you to leverage the power of TCL scripting to control and interact with the SystemVerilog environment.

4. Automation and Testbench Generation:

Automated testbench generation: Some tools can automatically generate UVM testbenches and associated TCL scripts for controlling the simulation process.

Tool integration: TCL can be used to integrate various EDA (Electronic Design Automation) tools within the verification flow, creating a seamless and automated workflow.

In essence, TCL serves as a valuable scripting language and command interface within the SystemVerilog UVM verification flow. It allows for simulation control, debugging, automation, and potentially testbench development, often in conjunction with the simulator's built-in capabilities and through the use of DPI and C interfaces.

References:

https://www.eetimes.com/using-tcl-to-create-a-virtual-component-in-verilog/?_ga

https://asic4u.wordpress.com/2018/05/05/uvm-forcing-signals-in-uvm-style/

https://vhdlwhiz.com/tcl-driven-testbench-for-vhdl-code-lock-module/

https://dvtalk.me/2020/10/18/forcing-signals-with-QuestaSim/

https://verificationacademy.com/forums/t/how-to-access-a-dut-signal-from-a-uvm-test-case-class/29099

https://verificationacademy.com/forums/t/how-to-force-dut-internal-signals-in-uvm-environment/28770

Python <-> TCL Interaction

How Python Can Interact with TCL Scripts?

Python can interact with Tcl scripts in several ways, mainly by leveraging built-in modules or external libraries
.

1. Using the tkinter Module:

Calling Tcl from Python: The tkinter module, primarily known for GUI development with Tk, includes a full-featured Tcl interpreter.

You can create a Tcl interpreter instance using tkinter.Tcl().

This instance provides methods like eval() and call() to run Tcl commands directly from Python.

eval() takes a single string containing the Tcl command and its arguments, returning the evaluated result as a string.

call() takes the Tcl command and arguments as separate arguments, returning the result as a corresponding Python data type.

Example:
python

```
import tkinter

tcl_interpreter = tkinter.Tcl()
result = tcl_interpreter.eval('expr 12 + 23')
print(result)    # Output: 35
```

Calling Python from Tcl: You can register Python functions as Tcl commands using the register() method of the Tcl interpreter instance.

2. Utilizing the subprocess Module:

Executing Tcl scripts as external processes: The subprocess module is a powerful tool to run external programs from Python.

You can use subprocess.run() to execute a Tcl script using the tclsh interpreter.

This approach is useful when you want to run standalone Tcl scripts and capture their output or handle return codes.

Example:
python

```
import subprocess

try:
    result = subprocess.run(['tclsh', 'your_script.tcl'], capture_output=True, text=True, check=True)
    print("Tcl script output:", result.stdout)
except subprocess.CalledProcessError as e:
    print("Error executing Tcl script:", e)
```

3. Third-party Libraries:

Tohil: This library provides bidirectional interaction between Tcl and Python, allowing you to call Python functions from Tcl and vice versa.

Elmer: This library allows developers to write code in Python and execute it within a Tcl environment, making Python code appear as native Tcl commands.

Choosing the right method:

The best approach depends on your specific needs:

If you need close integration, including direct execution of Tcl commands within your Python script and bidirectional function calls, using the tkinter module or dedicated libraries like Tohil is recommended.

If you simply need to execute a Tcl script as an external process and manage its input/output, the subprocess module is a suitable choice.

Python <--> TCL bidirectional

Python-Tcl-Interactions

https://wiki.tcl-lang.org/page/Accessing+Tcl+and+Python+from+one+another

https://wiki.tcl-lang.org/page/Python%2DTcl%2DInteractions

Tkinter

The de factor standard GUI library for Python, Tkinter, contains a full-featured Tcl interpreter, together with the Tk GUI toolkit. This allows running Tcl commands from Python, as well as Python commands from Tcl after performing

the some setup.

Tkinter is included with standard Linux, Windows and Mac OS X installations of Python

libtclpy

What libtclpy
Where https://github.com/aidanhs/libtclpy
https://core.tcl-lang.org/jenglish/gutter/packages/libtclpy.html
Description A Tcl extension to effortlessly to call bidirectionally between Tcl and Python, targeting Tcl >= 8.5 and Python 2.6 - 2.7.

tohil

https://github.com/flightaware/tohil
https://flightaware.github.io/tohil-docs/tutorial/tohil_tcl.html
https://pypi.org/project/tohil/
Tohil is inspired by libtclpy. It is simultaneously a Python extension and a Tcl extension that makes it possible to effortlessly call bidirectionally between Tcl and Python

tclpy

https://github.com/enics-labs/tclpy

https://deepwiki.com/The-OpenROAD-Project/OpenROAD/3.3-tcl-and-python-apis

Call From TCL to Python

https://wiki.tcl-lang.org/page/Python

https://www.tcl-lang.org/man/tcl8.5/tutorial/Tcl26.html

tclpython

https://wiki.tcl-lang.org/page/tclpython
What tclpython
Where https://github.com/amykyta3/tclpython/
Description Tcl extension which allows you to create python interpreters from within a Tcl application and evaluate python code.

Elmer

https://wiki.tcl-lang.org/page/Elmer
Elmer goes the other way. Elmer allows developers to write code in Python and execute it in Tcl. The resulting Tcl interface to the Python code generated by Elmer is transparent to the Tcl user... Python calls appear as Tcl calls ("foo(1, "a")" in Python appears as "foo 1 a" in Tcl, for example) and Python and Tcl data types are automatically mapped (Tcl lists are converted to Python lists, Python dictionaries are returned as Tcl associative arrays, etc.). Elmer also supports Python's "freeze" module, allowing a Python developer to deliver a single library consisting of several Python files "frozen" in to the Tcl application...no need to set PYTHONPATH or have Python source files accompanying the Tcl application.
tclpython

more···

https://gist.github.com/prideout/2766158

https://groups.google.com/g/comp.lang.tcl/c/U-KvgHNEoDQ

==========

https://community.altair.com/discussion/60426/calling-a-python-script-from-tcl-script/p1

https://gidsimulation.atlassian.net/wiki/spaces/GCS/pages/2816704556/3.+Using+Python+From+Tcl

https://adaptivesupport.amd.com/s/question/0D54U000067pqBPSAY/how-to-run-the-system-python-from-tcl-script-in-vivado-20222?language=en_US

https://community.foundry.com/discuss/topic/96228/python-wrapper-using-tcl

https://portwooddigital.com/2021/06/20/tcl-as-a-front-end-for-python/

https://www.reddit.com/r/FPGA/comments/ta46q9/how_to_run_python_scripts_from_vivado_tcl/

https://stackoverflow.com/questions/6825546/how-to-run-python-scripts-using-tcl-exec-command

https://community.unix.com/t/how-to-run-python-script-from-a-tcl-script/189288

Call From Python to TCL

https://wiki.tcl-lang.org/page/Python-Tcl-Interactions

tkinter

https://docs.python.org/3/library/tkinter.html

https://docs.python.org/3.9/library/tkinter.html?highlight=tkinter
https://docs.python.org/id/3.9/library/tkinter.html

tclwrapper

https://github.com/csail-csg/tclwrapper
https://pypi.org/project/tclwrapper/

notcl

https://notcl.readthedocs.io/en/latest/index.html
https://notcl.readthedocs.io/en/latest/

Typcl

Typcl is an unmaintained extension to use Tcl from Python
http://web.archive.org/web/20160911155836/http://equi4.com/critlib/typcl.README

python subprocess

https://stackoverflow.com/questions/14264118/running-tcl-through-python-by-subprocess-but-not-giving-any-output

more…

https://stackoverflow.com/questions/16439936/how-to-call-tcl-procedure-using-python

https://stackoverflow.com/questions/2519532/example-for-accessing-tcl-functions-from-python

https://stackoverflow.com/questions/3926273/can-you-embed-an-tcl-script-in-bash-script-or-python-script-thats-callable-by-e

https://stackoverflow.com/questions/7362846/running-tcl-code-from-python
https://stackoverflow.com/questions/7362846/running-tcl-code-from-python/27798805

https://stackoverflow.com/questions/57910326/running-tcl-script-from-python-with-arguments

==========

https://community.altair.com/discussion/63273/launching-tcl-scripts-with-arguments-from-python/p1

https://www.daniweb.com/programming/software-development/threads/433076/using-python-to-execute-a-tcl-command

https://community.foundry.com/discuss/post/1181387

Simulator commands

How are simulation scripts look like and how they can be written. Here I provide you with some links where you can find more info.

https://github.com/artvvb/digilent-vivado-scripts

TCL simulator commands

https://quartustcl.readthedocs.io/en/stable/
https://quartustcl.readthedocs.io/en/latest/
https://pypi.org/project/quartustcl/

Vivado Design Suite User Guide Using Tcl Scripting (UG894)

Python simulator commands

https://github.com/paulscherrerinstitute/VivadoScripting

https://adaptivesupport.amd.com/s/question/0D52E00006hpL1DSAU/vivado-automation-using-python?language=en_US

Python <--> TCL Converter

Of course, you may also convert TCL directly into Python in order to invoke your AI/ML modules written in Python.

However, as of now, it seems there is no easy tool to convert between Python and TCL straight away. Even if you use some tools and online platforms, manual modifications are mandatory afterwards. Here I have only listed some links where you may find some kind of conversion solutions, but not complete nor perfect.

Python-to-TCL

https://www.codeconvert.ai/python-to-tcl-converter

https://products.codeporting.ai/convert/python-to-tcl/

TCL-to-Python

https://www.codeconvert.ai/tcl-to-python-converter

https://products.codeporting.ai/convert/tcl-to-python/

https://www.thecodingforums.com/threads/converting-from-tcl-tkl-to-python.869203/

tcl2py

https://parflow.readthedocs.io/en/v3.13.0/python/tutorials/tcl2py.html
https://parflow-docs-update.readthedocs.io/en/stable/tutorials/tcl2py.html

https://git.acem.ece.illinois.edu/lib/VTK-5.0.0/src/commit/79f04fdc92bb3f88ecf79496e53ed7784ffb9c2d/Utilities/tcl2py.py

https://portwooddigital.com/2021/12/26/opensees-tcl-to-python-converter/

https://www.quora.com/Is-there-a-converter-that-will-help-me-convert-a-TCL-script-to-Python-or-a-way-to-approach-this-problem-of-mine