How to Customize the ModelSim Wave View in the Xilinx ISE Simulation Flow

Cristian Sisterna

Summary

When ModelSim is automatically lunched within the ISE environment it just displays the top entity level signals in the Wave View window. However, to either facilitate debugging tasks or check specific behavior of lower level components most of the time internal signals also need to be displayed in the Wave View window of ModelSim. Moreover, coloring, ordering and grouping signals is especially useful in complex designs. Hence, having one or several custom views and invoking them automatically will help the verification job.

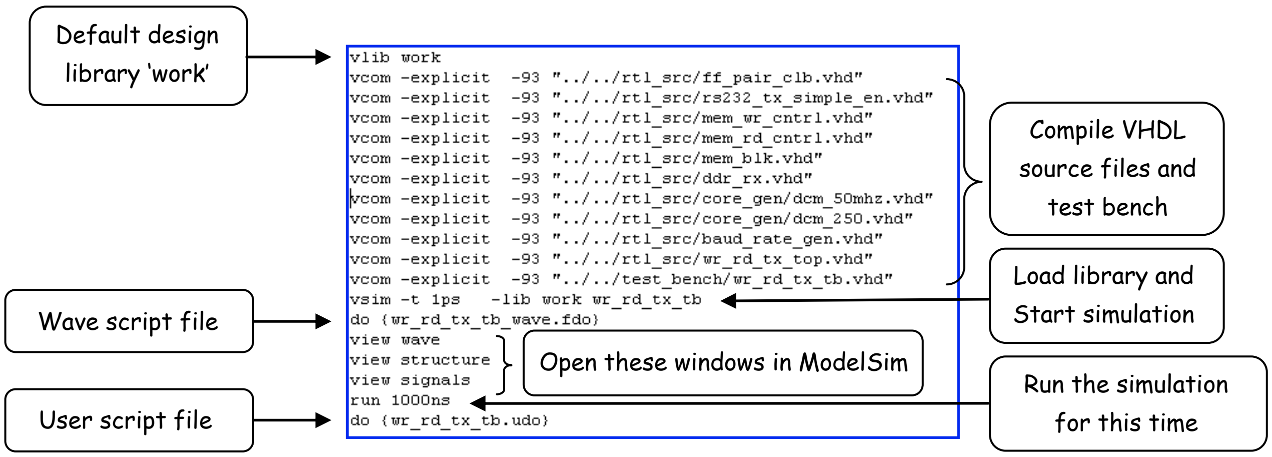
Scripts files (.do) created in ISE

When ModelSim is launched from the Xilinx ISE, Project Navigator automatically creates a `.do` script file that contains all of the necessary commands to compile the .vhd source files, load, start and run the simulation in ModelSim.

The `.do` script file created by Project Navigator has different extensions based on the type of simulation launched. For instance, for a Behavioral (functional) Simulation these are the three files created:

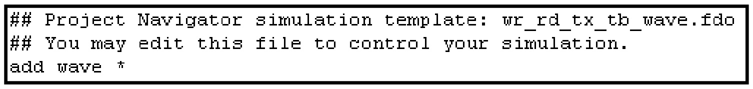
`<TestBenchName>.fdo`	ModelSim commands for behavioral simulation
`<TestBenchName>.udo`	ModelSim user commands
`<TestBenchName_wave>.fdo`	ModelSim Wave window format commands for behavioral simulation

A detail of the ModelSim commands in a `<TestBenchName>.fdo` script file created by Project Navigator, when running a Behavioral Simulation, is described below:


Default design library 'work'
Wave script file
User script file
Compile VHDL source files and test bench
Load library and Start simulation
Open these windows in ModelSim
Run the simulation for this time

```
vlib work
vcom -explicit  -93 "../../rtl_src/ff_pair_clb.vhd"
vcom -explicit  -93 "../../rtl_src/rs232_tx_simple_en.vhd"
vcom -explicit  -93 "../../rtl_src/mem_wr_cntrl.vhd"
vcom -explicit  -93 "../../rtl_src/mem_rd_cntrl.vhd"
vcom -explicit  -93 "../../rtl_src/mem_blk.vhd"
vcom -explicit  -93 "../../rtl_src/ddr_rx.vhd"
vcom -explicit  -93 "../../rtl_src/core_gen/dcm_50mhz.vhd"
vcom -explicit  -93 "../../rtl_src/core_gen/dcm_250.vhd"
vcom -explicit  -93 "../../rtl_src/baud_rate_gen.vhd"
vcom -explicit  -93 "../../rtl_src/wr_rd_tx_top.vhd"
vcom -explicit  -93 "../../test_bench/wr_rd_tx_tb.vhd"
vsim -t 1ps   -lib work wr_rd_tx_tb
do {wr_rd_tx_tb_wave.fdo}
view wave
view structure
view signals
run 1000ns
do {wr_rd_tx_tb.udo}
```

The `<TestBenchName>.fdo` script file is regenerated each time a behavioral simulation is launched. Therefore, any modification done in this file will be automatically overwriting. Still, in this script file there is a command line, `do{<TestBenchName_wave>.fdo}`, that invokes the `<TestBenchName_wave>.fdo` script file. This script file never is neither modified nor overwritten by Project Navigator. Hence, the best file that can be used to customize the Wave window format in the ModelSim environment is the `<TestBenchName_wave>.fdo` script file. By default, this file has the following structure:



```
## Project Navigator simulation template: wr_rd_tx_tb_wave.fdo
## You may edit this file to control your simulation.
add wave *
```

The '`add wave *`' command means that, by default, all the top level entity signals will be displayed in the Wave view of ModelSim.

Likewise, when running simulation after translating, mapping or PAR processes, similar script files are created. A detail of the script files created in each process is showed below:

Simulation Post-translate

`<TestBenchName>.ndo`	ModelSim commands for post-translate simulation
`<TestBenchName>.udo`	ModelSim user commands
`<TestBenchName_wave>.ndo`	ModelSim waves format commands for post-translate simulation

Simulation Post-Map Process

`<TestBenchName>.mdo`	ModelSim commands for post-map simulation
`<TestBenchName>.udo`	ModelSim user commands
`<TestBenchName_wave>.mdo`	ModelSim waves format commands for post-map simulation

Simulation Post-PAR

`<TestBenchName>.tdo`	ModelSim commands for post-par simulation
`<TestBenchName>.udo`	ModelSim user commands
`<TestBenchName_wave>.tdo`	ModelSim waves format commands for post-par simulation

In general, the script file `<TestBenchName_wave>.*do` will be the file to be modified to customize the Wave window in the ModelSim environment in any of the different simulation cases. As it was detailed above, the "`add wave *`" command states to display just the top level entity signals. However; to facilitate debugging tasks most of the time internal signals also need to be displayed in the Wave window. Moreover, coloring, ordering and grouping the signals in the Wave window is especially useful in complex designs.

Customization Process

The process to customize the Wave window in ModelSim when running the simulation flow in Xilinx ISE environment is detailed below:

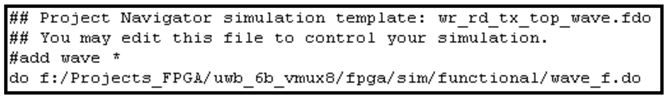
1. Create a new ISE project or just open one already created. Write your VHDL code, create your test bench and execute either the functional, post-translate, post-map or post-par simulation of your design, which will open ModelSim simulation tool.

2. Within the Wave window, ModelSim offers several methods to customize the view.
    a. *Adding internal signals:* in the Instance view of ModelSim (left most pane) you can find the test bench name and underneath it, you can see the instance name of the entity under test. The instance name is actually the label used in the instantiation of the top level entity in the test bench. Click over the '+' symbol to be able to see all the unit components of the top level entity. Then, select the unit that holds the signals you want to add to the Wave window by single click over the unit's name. After that, on the Objects window (usually the window in the middle) the name of the signals of the selected unit will be displayed. Select the signal you want to add to the Wave window by single click over it, then drag and drop the signal into the Wave window. You can repeat this process and add as many signals as needed. Once a signal is added to the Wave window, you can move it up or down as explained below.
    b. *Moving signals*: it is also possible to move either up or down the signals displayed in the Wave window to facilitate reading the waveforms. For instance, you can place all the input signals on the top of all the other signals, then the control signals in the middle and the output signals at the bottom. To move up or down the signals in the Wave window, select the signal you want to move (single right-mouse click over the signal). Then, drag the signal up or down, by keeping the right-mouse button pressed and moving the mouse up or down, and drop the signal, release the mouse button, wherever you like.
    c. *Adding dividers*: signal dividers can be added in the Wave window to label the signals grouped with a specific purpose. For instance, you can add dividers labeled Input Signals, Control Signals, Rx Signals, Memory Read, Clocks, and such. To add a divider select the signal over which you want the divider. Then, right-mouse click and select 'Insert Divider' from the down menu. The divider dialog window will come up. You can add a name describing the function of the signals that will be under the divider. You can also set the Divider Height, though this is not usually changed.
    d. *Coloring the waveform*: for some specific purposes, for instance for a quick waveform reference among hundreds of waveforms, it is advantageous to change the color of a specific waveform as well as the color of signal name. Highlight (select) the signal that you want to change the waveform color, then right-mouse click and select Properties, the Wave Properties window should come up. In the View tab, you will find the Wave Color and Name Color selections. To change the default colors, click on the Colors button and select the desired color. When finished, click on Apply.

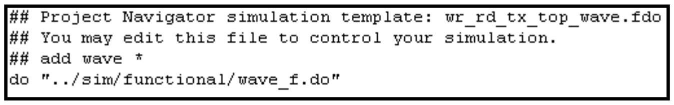
3. Once you are done adding signal, dividers and changing colors, it is then necessary to save this new customized Wave window format to be able to use it again:
    a. Be sure to select (highlight) the Wave window among the other ModelSim views.
    b. Go to File -> Save Format. Then, click OK in the Save Format dialog window to save the new wave format in the default directory (ISE project directory) or in the directory you would like by browsing to it. The default name is wave.do. In case you are planning to have a customized wave view for the different simulations (like functional, post-map, post-PAR), change the default wave.do name to something like wave_f.do for functional, wave_m.do for mapping and so on for the other type of simulation.

4. Go back to the ISE Project Manager window.

5. There are a couple of options to load automatically the saved wave.do file (item 3.b) when running a simulation from Project Navigator. As it was explained before, the easiest way is to modify the `<TestBenchName_wave.fdo>` file. Hence, open to edit the `<TestBenchName_wave.fdo>` file that resides in the ISE Project Directory. Remove or comment out (by using the # character) the command line "`add wave *`" and then add a new command line that will invoke the customized waveform; by writing "`do wave.do`" (or whatever name you use when saving the Wave window format). If you have saved the wave.do file in a directory different than the ISE Project Directory, you will need to write the complete path where the file `wave.do` resides. Below, there are two examples of how the `<TestBenchName_wave.fdo>` file would look like, one for a complete directory path, and the other with a relative path:



```
## Project Navigator simulation template: wr_rd_tx_top_wave.fdo
## You may edit this file to control your simulation.
#add wave *
do f:/Projects_FPGA/uwb_6b_vmux8/fpga/sim/functional/wave_f.do
```



```
## Project Navigator simulation template: wr_rd_tx_top_wave.fdo
## You may edit this file to control your simulation.
## add wave *
do "../sim/functional/wave_f.do"
```

Once finishing with the modifications, save the customized `<TestBenchName_wave.fdo>` file.

6. From now on for every behavioral simulation to be executed, Project Navigator will use the custom wave.do waveform format when launching ModelSim.

Even though it has been explained the process to customize the behavioral simulation, similar steps have to be followed, for instance, for customizing the post-place and route simulation. The file to be changed in this case is the `<TestBenchName_wave>.tdo` (as it was explained before) and just follows all the steps detailed above for the behavioral simulation.

More on Customization

Besides of modifying the `<TestBenchName_wave.fdo>` file for customizing the Wave window format, another ModelSim commands can also be added with other purposes. For instance, one useful command for designs with a large amount of internal signals is the following:

```
log -r /*
```

This command will log *all* the data objects in the design. For example, if after running the simulation you find out you would like to see an internal signal not currently displayed in the Wave view, you just need to select the signal you want to add, drag and drop it in the Wave view. Then, its respective waveform will immediately be displayed. Without the `log` command, if that particular internal signal is not in the list of signals in the wave.do script, it is not logged; therefore no waveform will be displayed for that signal. In this case, it will be necessary to add the signal to the Wave window, save the wave.do again and then rerun the simulation. The drawback of the log command is that it could make the simulation much slower since it needs to log *all* the signals of the design.

One point to keep in mind is the fact that internal signals after PAR usually do not keep the same name as before the PAR. This means that the wave.do saved for functional simulation, may or may not be able to display all the internal signals when doing a post-PAR simulation. One solution for this problem is to create another customized wave.do for the post-PAR simulation naming it something like `wave_par.do`. Then, change the `<TestBenchName_wave.tdo>` file as explained in point 5 above.

Another helpful point to know is the fact that it is possible to rerun the simulation without closing ModelSim, avoiding returning to Project Navigator. For instance, if you find an error when running the simulation in one of the .vhd source files, you can edit the VHDL file, still keeping open ModelSim (even you can use ModelSim text editor), save the modifications of the VHDL file and rerun the simulation by typing `do {TestBenchName.fdo}` and pressing enter in the transcript window (bottom window) of ModelSim. You can also use the up-arrow key in the transcript window to go through all the executed commands until you find the `do {TestBenchName.fdo}` command, and then just press enter to execute it.