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

Cristian Sisterna

## Summary

When ModelSim is automatically lunched from the ISE environment it just displays the top level entity 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 panel of ModelSim. Moreover, coloring, ordering and grouping signals is especially useful when simulating complex designs. Therefore, having one or several custom views and invoking them automatically will help the verification job. This application note details the necessary steps to setting up the ISE environment, as well as the ModelSim commands to open a customized Wave View window panel automatically.

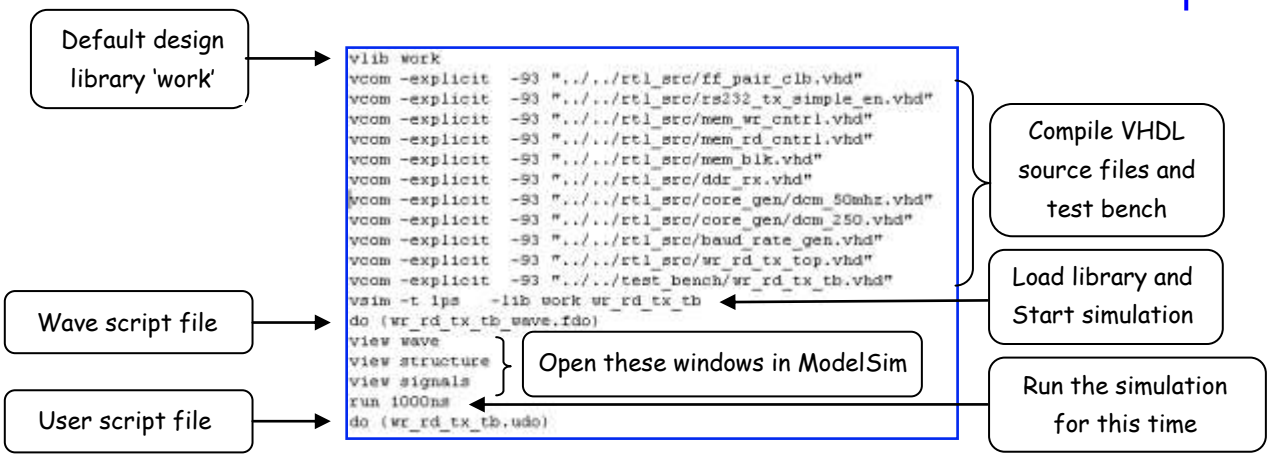
## Scripts files (.do) created by ISE

When ModelSim is launched from the Xilinx ISE environment, Project Navigator automatically creates a `.do` script file that contains all of the necessary commands to compile the .vhd source files and the test bench, 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:




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. However, in this script file there is a command line, `do{<TestBenchName_wave>.fdo}`(see the figure above) that invokes the `<TestBenchName_wave>.fdo` script file. This script file is neither modified nor overwritten by Project Navigator. Hence, the best file that can be used to customize the Wave View window panel 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 all the top level entity signals will be displayed in the Wave View window panel of ModelSim.

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

## Scripts for Post-translate Simulation

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

## Scripts for Post-Map Process Simulation

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

## Scripts for Post-PAR Simulation

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

Generally, the script file `<TestBenchName_wave>.*do` will be the file to be modified to customize the Wave View window panel 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.

## Waveform Customization Process

The different steps to follow to be able to customize the Wave View window panel in ModelSim when running the simulation flow in the Xilinx ISE environment are:

1. Create a new ISE project and write the necessary VHDL code or just open an existing project

2. If the test bench has already been written go to step 3, otherwise write the respective test bench and add the file to the project.

3. Execute either the functional, post-translate, post-map or post-par simulation of the design. Then, the respective script detailed above is generated and executed. Thus, ModelSim is invoked and will come up to run the simulation as defined in the test bench and will open the window panels that are detailed in the simulation script just created, by default they are: Wave Window, Structure and Signals.

4. After the simulation stops, keep ModelSim open. The following steps will detail the customization process of the Wave View window panel.

5. ModelSim offers several methods to customize the view in the Wave View window panel. In this step the most commonly used techniques will be described and detailed to later add them to the customized script file:
    a. *Adding internal signals:* to add internal signals to the Wave View window goes to the Instance window panel of ModelSim (left most panel) and find the test bench name. Underneath the test bench find the component instantiation label of the entity under test. This label is the label used in the component instantiation of the top level entity in the test bench code. It usually is something like UUT, U1, DUT or something similar. Click over the '+' symbol on the left of the instantiation label to expand the component instantiations and be able to see all the components of the top level entity. Then, single click over the component (design unit) that holds the signal(s) you want to add to the Wave View window panel. Hence, the name of all the signals of the selected design unit will be displayed on the Objects window panel (usually the window in the middle of ModelSim). In case the Objects window panel is not visible, go to the bar menu and select View -> Objects. Then, single click on the signal to add to the Wave View window panel and then drag and drop the signal into the Wave View window panel. You can also do a single click over the signal, or keep Ctrl pressed and click on all the signals you want to

add, then right click and select Add->ToWave->Selected Signals. Repeat this process and add as many signals as needed. Once a signal is added to the Wave View 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 View 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 part of the waveforms and the output signals at the bottom. To move up or down the signals in the Wave View window panel, select the signal you want to move (single left-mouse click over the signal). Then, drag the signal up or down, by keeping the left-mouse button pressed and moving the mouse up or down, and drop the signal, wherever you like, by releasing the mouse button. You can select a group of signals to move up or down by using the Shift or Control keys as you usually use them.

c. *Adding dividers*: signal dividers can be added in the Wave View window panel 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 pull 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. To change the color, first highlight (select) the name of the signal that you want to change the color, then right-mouse click and select Properties. The Properties window should come up. In the View tab, you will find the Wave Color and Name Color options. To change the default color of either the Wave Color or the Name Color, click on the respective Colors button and select the desired color. When finishing, click on Apply.

6. Once you are done adding signal, dividers and changing colors, it is then necessary to save this new customized Wave View window panel 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. The Save Format window will come up. The default name for the wave format 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 simulations. 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 to use by browsing to it.

7. Close ModelSim and go back to the ISE Project Manager window.

8. Once the new format of the Wave View window panel has been saved either as `wave.do` or as your customized `.do` file, there are a couple of options to load automatically the saved script file (*.do) (step 6.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 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 View window panel 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.

9. From now on for every simulation to be executed, ModelSim will use the custom `wave.do` waveform format.

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. In this case, the file to be changed 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 View window panel format, another ModelSim commands can also be added to the script file with other purposes. For instance, one useful command for designs with a large amount of internal signals is the following:

```
log -r /*
```

This ModelSim command will log *all* the data objects in the design. This is very useful when, for instance, after running the simulation you find out that you would like to see an internal signal not currently displayed in the Wave View window panel. By using `log -r /*`, 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 View 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.

On the other hand, one point to keep in mind when dealing with the signals added to the script file(s) is the fact that the internal signals of the design after PAR usually do not keep the same name as before the PAR. This means that the `wave.do` saved for functional simulation, might not be able to display all the internal signals added to the script when doing a post-place and route simulation when doing a post-PAR simulation. A 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 8.

Another helpful point to be familiar with 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.